

DATABASES AND INFORMATION SYSTEMS VI

VISIT...

LANZAROTE
Caliente.COM

Frontiers in Artificial Intelligence and Applications

FAIA covers all aspects of theoretical and applied artificial intelligence research in the form of monographs, doctoral dissertations, textbooks, handbooks and proceedings volumes. The FAIA series contains several sub-series, including “Information Modelling and Knowledge Bases” and “Knowledge-Based Intelligent Engineering Systems”. It also includes the biennial ECAI, the European Conference on Artificial Intelligence, proceedings volumes, and other ECCAI – the European Coordinating Committee on Artificial Intelligence – sponsored publications. An editorial panel of internationally well-known scholars is appointed to provide a high quality selection.

Series Editors:

J. Breuker, N. Guarino, J.N. Kok, J. Liu, R. López de Mántaras,
R. Mizoguchi, M. Musen, S.K. Pal and N. Zhong

Volume 224

Recently published in this series

- Vol. 223. R.G.F. Winkels (Ed.), Legal Knowledge and Information Systems – JURIX 2010: The Twenty-Third Annual Conference
- Vol. 222. T. Ágotnes (Ed.), STAIRS 2010 – Proceedings of the Fifth Starting AI Researchers’ Symposium
- Vol. 221. A.V. Samsonovich, K.R. Jóhannsdóttir, A. Chella and B. Goertzel (Eds.), Biologically Inspired Cognitive Architectures 2010 – Proceedings of the First Annual Meeting of the BICA Society
- Vol. 220. R. Alquézar, A. Moreno and J. Aguilar (Eds.), Artificial Intelligence Research and Development – Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence
- Vol. 219. I. Skadiņa and A. Vasiljevs (Eds.), Human Language Technologies – The Baltic Perspective – Proceedings of the Fourth Conference Baltic HLT 2010
- Vol. 218. C. Soares and R. Ghani (Eds.), Data Mining for Business Applications
- Vol. 217. H. Fujita (Ed.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the 9th SoMeT_10
- Vol. 216. P. Baroni, F. Cerutti, M. Giacomini and G.R. Simari (Eds.), Computational Models of Argument – Proceedings of COMMA 2010
- Vol. 215. H. Coelho, R. Studer and M. Wooldridge (Eds.), ECAI 2010 – 19th European Conference on Artificial Intelligence
- Vol. 214. I.-O. Stathopoulou and G.A. Tsihrintzis, Visual Affect Recognition

ISSN 0922-6389 (print)
ISSN 1879-8314 (online)

Databases and Information Systems VI

Selected Papers from the Ninth International Baltic Conference,
DB&IS 2010

Edited by

Janis Barzdins

University of Latvia, Latvia

and

Marite Kirikova

Riga Technical University, Latvia

IOS
Press

Amsterdam • Berlin • Tokyo • Washington, DC

© 2011 The authors and IOS Press.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without prior written permission from the publisher.

ISBN 978-1-60750-687-4 (print)

ISBN 978-1-60750-688-1 (online)

Library of Congress Control Number: 2010941319

Publisher

IOS Press BV

Nieuwe Hemweg 6B

1013 BG Amsterdam

Netherlands

fax: +31 20 687 0019

e-mail: order@iospress.nl

Distributor in the USA and Canada

IOS Press, Inc.

4502 Rachael Manor Drive

Fairfax, VA 22032

USA

fax: +1 703 323 3668

e-mail: iosbooks@iospress.com

LEGAL NOTICE

The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

Preface

The Ninth International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2010) took place on July 5–7 in Riga. This conference is continuing a series of successful biennial Baltic conferences on databases and information systems, which have been held in Trakai (1994), Tallin (1996, 2002, 2008), Riga (1998, 2004), and Vilnius (2000, 2006).

During this period, Baltic DB&IS conferences has become real international forums of high scientific criteria for academics and practitioners in the field of databases and advanced information systems (IS) as well as in related areas, such as semantic technologies, ontologies, advanced software engineering (SE) technologies for IS development, and IS and security.

Baltic DB&IS'2010 was organized by the Institute of Mathematics and Computer Science, University of Latvia, and the Faculty of Computing, University of Latvia, in co-operation with the Faculty of Computer Science and Information Technology, Riga Technical University.

The Programme Committee of Baltic DB&IS'2010 consisted of 35 members from 14 countries. 59 papers from 12 countries were submitted for the conference including 13 papers from 5 countries for Doctoral Consortium. Each conference paper was reviewed by three referees, as a rule, from different countries. As a result, 28 papers were accepted and presented at the conference. 27 revised and extended best papers are included in this volume.

The papers present original results concerning integration and security of IS, semantic technologies for IS, domain specific languages and tools for IS, SE with models and ontologies, model based data storages, and business IS.

We express our warmest thanks to all authors who contributed to the conference. Our special thanks to the invited speakers Prof. Hele-Mai Haav, Prof. Jürgen Ebert and Prof. Michal Smialek for sharing their knowledge in advanced systems development methods. We are grateful to members of the Programme Committee and the additional referees for a careful reviewing of submissions.

We would also like to thank all the organizing team and our sponsors, who have made this conference and publishing of this book possible. Our sincere gratitude to Conference Secretary Elina Kalnina for her immense technical work.

Last, but not least, we thank all participants who really made the conference.

October, 2010

Juris BORZOVS, Conference Chair
Janis BARZDINS, Programme Co-Chair
Marite KIRIKOVA, Programme Co-Chair

This page intentionally left blank

Conference Committee

Conference Chair

Juris Borzovs, Faculty of Computing, University of Latvia

Organising Co-Chairs

Guntis Arnicans, Faculty of Computing, University of Latvia

Inara Opmane, Institute of Mathematics and Computer Science, University of Latvia

Conference Secretary

Elina Kalnina, Institute of Mathematics and Computer Science, University of Latvia

Co-ordinators

Saulius Maskeliunas, Institute of Mathematics and Informatics, Lithuania

Hele-Mai Haav, Institute of Cybernetics at Tallinn University of Technology, Estonia

Ahto Kalja, Department of Computer Engineering of Tallinn University of Technology, Estonia

Advisory Committee

Janis Bubenko, Royal Institute of Technology, Sweden

Arne Solvberg, Norwegian University of Science and Technology, Norway

Programme Co-Chairs

Janis Barzdins, University of Latvia

Marite Kirikova, Riga Technical University

Programme Committee

Witold Abramowicz, Poland

Irina Astrova, Estonia

Guntis Barzdins, Latvia

Janis Bicevskis, Latvia

Juris Borzovs, Latvia

Albertas Caplinskas, Lithuania

Vytautas Cyras, Lithuania

Dale Dzemydiene, Lithuania

Johann Eder, Austria

Hans-Dieter Ehrich, Germany

Janis Grabis, Latvia

Hele-Mai Haav, Estonia

Janis Grundspenkis, Latvia

Mirjana Ivanovic, Serbia

Leonid Kalinichenko, Russia

Ahto Kalja, Estonia

Audris Kalnins, Latvia

Peep Küngas, Estonia

Rein Kuusik, Estonia

Marion Lepmets, Estonia

Audrone Lupeikene, Lithuania

Kalle Lyytinen, USA

Hui Ma, New Zealand
 Rainer Manthey, Germany
 Saulius Maskeliunas, Lithuania
 Jorgen Fischer Nilsson, Denmark
 Boris Novikov, Russia
 Algirdas Pakstas, UK
 Jaan Penjam, Estonia
 Karlis Podnieks, Latvia
 Gunter Saake, Germany
 Kurt Sandkuhl, Sweden
 Michal Smialek, Poland
 Darja Smite, Sweden
 Janis Stirna, Sweden

Additional Referees

Erika Asnina, Latvia
 Edgars Celms, Latvia
 Karlis Cerans, Latvia
 Boris Cogan, UK
 Janis Eiduks, Latvia
 Kārlis Freivalds, Latvia
 Rūsiņš Freivalds, Latvia
 Ingolf Geist, Germany
 Martin Henkel, Sweden
 Kristi Kirikal, Estonia
 Christian Koncilia, Austria
 Arne Koschel, Germany
 Deniss Kumlander, Estonia
 Julius Köpke, Austria

Uldis Sukovskis, Latvia
 Kuldar Taveter, Estonia
 Jaak Tepandi, Estonia
 Benhard Thalheim, Germany
 Enn Tyugu, Estonia
 Olegas Vasilecas, Lithuania
 Tatjana Welzer, Slovenia
 Wita Woitkowski, USA
 Robert Wrembel, Poland
 Stanislaw Wrycza, Poland
 Naoki Yonezaki, Japan
 Jozef M. Zurada, USA

Algirdas Laukaitis, Lithuania
 Egons Lavendelis, Latvia
 Azeem Lodhi, Germany
 Leonids Novickis, Latvia
 Martins Opmanis, Latvia
 Syed Saif ur Rahman, Germany
 Tarmo Robal, Estonia
 Eike Schallehn, Germany
 Inna Shvartsman, Estonia
 Renate Strazdina, Latvia
 Vladimir Tarasov, Sweden
 Sarah Tauscher, Germany
 Juris Viksna, Latvia

Doctoral Consortium Co-Chairs

Guntis Barzdins, University of Latvia
 Leo Selavo, University of Latvia

Local Organising Committee

Vineta Arnicane
 Edgars Celms
 Dainis Dosbergs
 Girts Kamitis
 Lelde Lace
 Inga Medvedis
 Edgars Rencis
 Agris Sostaks
 Viesturs Zarins

Contents

Preface	v
<i>Juris Borzovs, Janis Barzdins and Marite Kirikova</i>	
Conference Committee	vii
Invited Papers	
Ontology-Driven Development of Personalized Location Based Services	3
<i>Hele-Mai Haav, Aivi Kaljuvee, Martin Luts and Toivo Vajakas</i>	
Interoperability Services for Models and Ontologies	19
<i>Jürgen Ebert and Tobias Walter</i>	
Requirements-Level Programming for Rapid Software Evolution	37
<i>Michał Śmiałek</i>	
Tools, Techniques and Languages for Model Driven Development	
An Approach for Enacting Software Development Process: SPEM4MDA	55
<i>Vladimirs Nikulsins, Oksana Nikiforova and Jurijs Kornijenko</i>	
Bringing Domain Knowledge to Pattern Matching	66
<i>Agris Sostaks</i>	
A Kernel-Level UNDO/REDO Mechanism for the Transformation-Driven Architecture	80
<i>Sergejs Kozlovics, Edgars Rencis, Sergejs Rikacovs and Karlis Cerans</i>	
On Views on Metamodels	94
<i>Edgars Rencis</i>	
Associations as First-Class Elements	108
<i>Daniel Bildhauer</i>	
Semantic Technologies for Information Systems	
Multilevel Data Repository for Ontological and Meta-Modeling	125
<i>Mārtiņš Opmanis and Kārlis Čerāns</i>	
RDB2OWL: A RDB-to-RDF/OWL Mapping Specification Language	139
<i>Kārlis Čerāns and Guntars Būmans</i>	
Spatial Ontology in Factored Statistical Machine Translation	153
<i>Raivis Skadiņš</i>	

Domain Specific Languages and Tools

Practitioners View on Domain Specific Business Process Modeling <i>Janis Bicevskis, Jana Cerina-Berzina, Girts Karnitis, Lelde Lace, Inga Medvedis and Sergejs Nesterovs</i>	169
The Design of Electronic Service Process Using YAWL and CPN Tools <i>Peteris Stipravietis and Maris Zieme</i>	183
Grammatical Aspects: Coping with Duplication and Tangling in Language Specifications <i>Andrey Breslav</i>	197

Information Systems Integration

XML-Based Specification of the Project Management Domain and Its Application <i>Solvita Bērziša</i>	213
Combining Functional and Nonfunctional Attributes for Cost Driven Web Service Selection <i>Mārtiņš Bonders, Jānis Grabis and Jānis Kampars</i>	227
WSRF Usage for BPM and CEP Systems Integration <i>Normunds Blumbergs and Maksims Kravcevs</i>	240

Business Information Systems and Software Process Improvement

Value and Viability Considerations in Information Systems Development <i>Einar Polis</i>	257
L.O.S.T Records: The Consequence of Inadequate Recordkeeping Strategies <i>Erik A.M. Borglund and Karen Anderson</i>	271
Evaluation of the Archetypes Based Development <i>Gunnar Piho, Jaak Tepandi and Mart Roost</i>	283
Comparison of Plan-Driven and Agile Project Management Approaches: Theoretical Bases for a Case Study in Estonian Software Industry <i>Marion Lepmets and Margus Nael</i>	296
Test Points in Self-Testing <i>Edgars Diebelis and Janis Bicevskis</i>	309

Information Systems and Security

Integrating IT Governance, Risk, and Compliance Management Processes <i>Nicolas Racz, Edgar Weippl and Andreas Seufert</i>	325
Towards Model Transformation Between SecureUML and UMLsec for Role-Based Access Control <i>Raimundas Matulevičius and Marlon Dumas</i>	339

Information Security Management Method for Households <i>Ilze Murane</i>	353
Information Systems and AI Technologies	
Random Spikes to Enhance Learning in Spiking Neural Networks <i>Janis Zuters</i>	369
Interactive Inductive Learning System <i>Ilze Birzniece</i>	380
Two Player Fair Division Problem with Uncertainty <i>Andre Veski and Leo Vohandu</i>	394
Visualization of Airport Procedures in Time Critical Decision Support Systems <i>Kristina Lapin, Vytautas Čyras and Laura Savičienė</i>	408
Parallel Bidirectional Dijkstra's Shortest Path Algorithm <i>Gintaras Vaira and Olga Kurasova</i>	422
Subject Index	437
Author Index	439

This page intentionally left blank

Invited Papers

This page intentionally left blank

Ontology-driven Development of Personalized Location Based Services

Hele-Mai HAAV ^{a,1}, Aivi KALJUVEE ^b, Martin LUTS ^c and Toivo VAJAKAS ^b

^a*Institute of Cybernetics at Tallinn University of Technology, Estonia*

^b*Regio Ltd, Estonia*

^c*ELIKO Competence Centre in Electronics-, Info- and Communication Technologies, Estonia*

Abstract. Personalization is very important feature of any Location Base Service (LBS) as it improves its usability. The paper focuses a problem of development of personalized LBSs. In the paper, a novel approach based on ontology engineering is proposed to provide an intelligent (semantics-based) solution to personalization problem of LBS. The approach uses geospatial ontologies, ontology-based user profiling and multilingual output for personalization of services. The provided ontology-driven development framework for personalized LBS is evaluated by implementing the personalized reverse geo-coding service use case.

Keywords. development of LBS, personalized LBS, geo-spatial ontology, ontology verbalization, reverse geo-coding

Introduction

Location Based Services (LBS) [1] are becoming increasingly popular in the nearest future. However, development of LBS is not cost-effective due to semantic heterogeneity of geo spatial data sources and positioning systems. Currently, users require more personalized LBSs than before. This demand in turn adds complexity to LBS development process.

The paper focuses a problem of development of personalized LBSs. As a motivating use case an advanced personalized reverse geo-coding service is considered. It produces a natural language like description of the location of the user including spatial features (e.g., shops, hotels, etc) that are spatially relevant to it and meet the user preferences.

There are different aspects to take into account in order to personalize LBS as follows: location and time, user preferences related to spatial features and to spatial relationships, user preferences concerning the user interface of the LBS, etc.

In this paper, a novel approach based on ontology engineering is proposed to provide an intelligent (semantics-based) solution to personalization problem of LBS. Ontology based approach was chosen because it makes possible to overcome heterogeneity of different geo-databases, reuse domain knowledge over applications, and capture user preferences as well as application context. User preferences can be

¹ Corresponding Author. Institute of Cybernetics at Tallinn University of Technology, Akadeemia 21, 12618 Tallinn, Estonia; E-mail:helemai@cs.ioc.ee

represented and collected by using different methods and techniques [2]. One way is to use ontologies for capturing user interests and preferences [2, 3, 4] as also shown in this paper.

Location of the user is one of the most important aspects that we take into account. We do not consider actual time in our approach, but this could be easily added to the framework. We take into account the user preferences concerning spatial features and relationships as well as language. People express and understand spatial relations through natural language instead of metric measurement. Therefore, it is important to express imprecise spatial references (like “near to”, “in between”, “far”, etc) explicitly by ontologies when developing advanced personalized LBS. The presented solution aims at providing ontology based method for enriching spatial features and relationships with relevant to the user meaning in order to explicitly capture such imprecise spatial references.

Geographic Information System (GIS) is important component of any LBS. Geo-databases in general contain a lot of geographic information but these are usually optimized for printing maps (e.g. Regio Ltd databases) and as such are not sufficient for effective retrieval of spatial features as needed for LBSs. This was one of the reasons why we decided to create geospatial ontologies. These ontologies serve as a basis for representation of user preferences. Introduced spatial ontologies enable also cope with heterogeneity of geo-databases and other information sources.

For verbalization of the results of filtering of candidate spatial features and relationships, we have chosen Adaptive Natural Language Generation (ANLG) approach that can produce human understandable texts from some underlying non-linguistic representation of information [5]. We present a method for multilingual ontology verbalization applied to retrieved ontology instances presented in RDF.

The provided ontology-driven development approach for personalized LBS combines above described solutions in order to resolve a wider range of ontology-driven LBSs than mentioned reverse geo-coding service that is used for evaluation of the framework.

Novelty of our contribution is twofold; it makes LBS development independent on heterogeneity of data sources and enables personalization of LBS. Personalization is based by enriching purely topological spatial relationships with meaning that end users will expect and reasoning on such *semantic* spatial relationships. This is achieved due to applying ontologies in the development of LBSs.

The structure of the paper is as follows. Section 1 gives an overview of our motivating use case. Section 2 outlines the proposed architecture for development of ontology-based personalized LBS. Its most important subsystems of ontologies and personalization are considered in sections 3 and 4. Section 5 is devoted to verbalization of filtering results. Section 6 describes evaluation of the approach. Section 7 is devoted to related work and section 8 concludes the paper.

1. Use Case Description

Traditional reverse geo-coding service (RGCS) is a LBS, a process of backward coding of geographic position/coordinate into a readable address.

The input of the RGCS is a geographic coordinate or a position. The input may be determined by GPS, mobile cellular network, the user click within a GIS/spatially-

enabled application, or by other means. Different input-types have different accuracy characteristics.

Usually, the output of the RGCS is a human and/or computer readable address of the geographic position. Reverse geo-coding can also tell where is the nearest populated place or which administrative division the coordinate is placed in. An example of the RGCS API is a Google Maps API Client Geocoder [6].

In this paper, we are motivated by the situation, where the address as an output of the RGCS (for example, a name of a by-street) may not be any help to the user (the end-user might be for example in a foreign place/town). Another concern is that in many cases the nearest address-point may not be accurate, for example the house number may be interpolated. This calls for the development of an *advanced* RGCS, which enables the end-user to ask for a description of a geographical position (which may be carried out by sending a SMS to a RGCS number). The position may be the one the user is standing at, a click on a map, or other.

The output of LBS is required to be the textual description of relevant to the user recognizable objects (spatial features) together with their spatial relationships related to the location of the user. The output may be accompanied with a map, highlighting these objects. The provision of this additional feature depends on the capabilities of the device of the end-user.

The motivating use case is initiated by Regio Ltd, the leading mapping agency of Estonia. Data sources needed for the current development of LBS implementing the case study are provided by Regio Ltd, who maintains a database of the topography of Estonia. The database contains a large number of spatial features representing towns, forests, roads, rivers, individual buildings, etc.

In general, the provided solution is independent on data source as mapping to different GIS databases could be done via data ontology as described in this paper.

2. Overview of the Approach

In order to implement this use case and other personalized LBS integration of several heterogeneous data sources, domain knowledge as well as knowledge about user preferences and context of LBS is needed.

For capturing this knowledge we propose ontology based solution – Smart Semantic Space FrameWork, abbreviated as S3FW – which will cover the reverse geocoding use case described in section 1, but also resolve a wider range of ontology-driven LBSs and geo-services: routing, positioning, route finding, navigation, parsing and understanding spatial queries among them. The S3FW is component based, allowing us to implement it incrementally on use-cases demand, but still having an extendable, visionary approach.

We have designed S3FW as an additional, “semantic” module to be used as an add-on to proprietary GIS products (see Figure 1). A proprietary GIS solution needs to be modified to have additional hooks in its business layers to S3FW, which offers ontology-based services like classification of spatial objects, multilingual verbalization of them, etc.

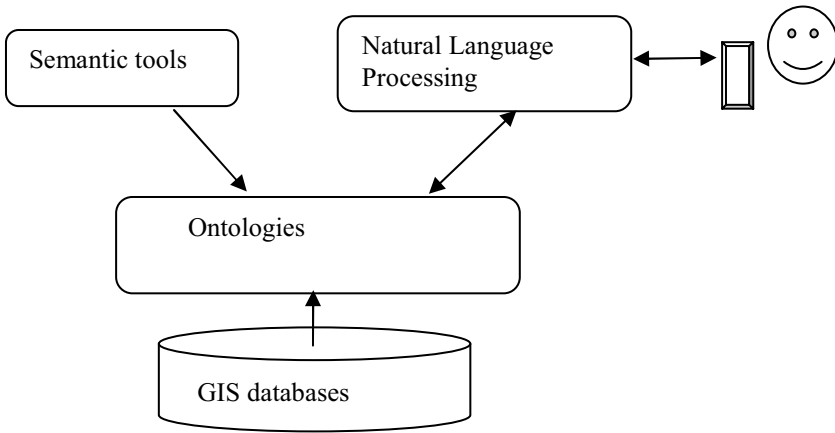


Figure 1. Ontology-driven LBS development architecture S3FW

In the architecture of S3FW (see Figure 1), there are the following important subsystems and major components:

- a subsystem of ontologies, fulfilling the following roles:
 - expressing formalized, explicit knowledge about the domain of interest, for example taxonomies of Point of Interests (POIs), user profile ontologies, ontology for spatial relationships, etc;
 - acting as a linguistic database for multilingual labels of terms about the domain of interest, for example classes of POIs, to be used in verbalization component;
- a subsystem of semantic tools including description logic (DL) reasoners that perform the following tasks;
 - verification of consistency of ontologies;
 - automatic classification of ontology instances;
- a subsystem of Natural Language Processing (NLP) performing the following tasks:
 - Natural Language Understanding (NLU), to be used to parse and understand queries given in (controlled) natural languages;
 - Natural Language Generation (NLG), to be used to produce natural-language-alike outcomes of geo-services, for example: RGCS.

3. S3FW Domain Ontologies

3.1. Goal and Scope of Ontologies

Main *goal* of ontologies in the S3FW is to give an explicit and formal meaning to the geospatial domain (content) concepts. Another important goal is to express user preferences by using concepts from the geospatial domain ontology.

The *scope* of the ontologies developed for S3FW is not limited to the reverse geocoding service described above, but most of these ontologies can be reused and extended for other geospatial applications and LBS.

Ontologies of the S3FW are processed by machines. However, human-friendly explanation of ontological terminology in different languages is given in annotations and labels of a particular ontology class or property. This is also used for multilingual ontology verbalization as described in this paper. OWL DL [7] was chosen for the S3FW ontology representation as it is formal, decidable and expressive enough.

This paper gives a brief overview of ontology component of the S3FW. Detailed discussion of ontology architecture could be found in [8]. The current paper concentrates on how to apply S3FW ontologies for personalization of LBS rather than presenting all design details of ontologies used.

3.2. Ontology Structure

The need to use ontologies for representing semantics of LBSs has been growing in recent years and several ontology structures have been proposed for LBSs domain [9, 10, 11, 12]. For example, according to [12], space, time and movement of objects are the common concepts in the field of LBS. They provide ontology architecture for LBS consisting of domain, content and application ontologies. In their work, the domain ontology is a top ontology for the LBSs capturing only meaning of generic concepts like location, time, movement and others related to all LBSs. In the LBS application level, they talk about content, use profile and service ontologies as these depend on the specific application.

In relation to the approach described above, we designed different architecture of ontologies as our starting points were geo-domain ontologies and GIS related databases. In terms of their work, our ontology architecture is on LBS application level. The concept of location (space) is introduced to our ontology structure via concept of geometry of spatial features as well as by representing spatial relationships among features.

Figure 2 presents the architecture of ontologies for the ontology-driven LBS development framework S3FW.

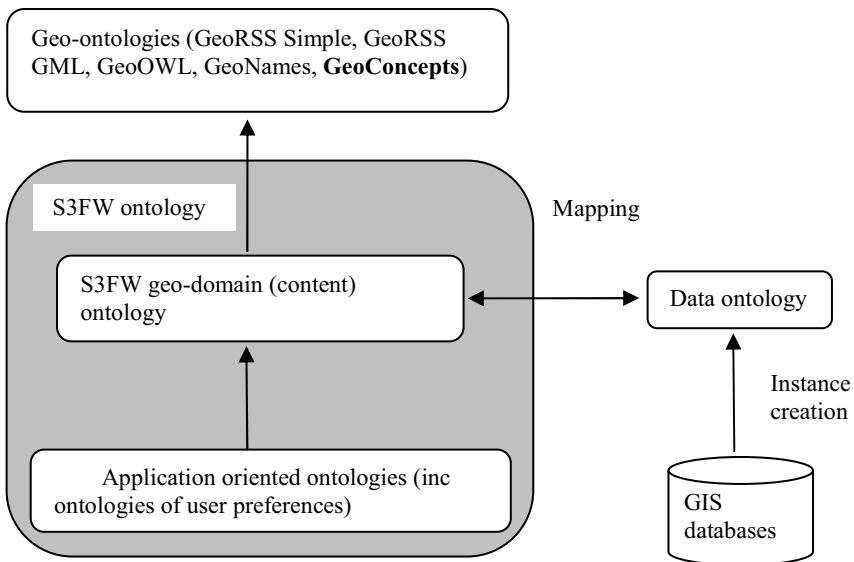


Figure 2. Architecture of S3FW ontology

The S3FW ontology reuses one of the existing geo-ontology GeoConcepts [13] that will be described later in this paper. The original ontology was divided into two separate modules: S3FW geo-domain ontology and application oriented ontologies that include user preferences ontology. This division enables easy reusing the domain ontology. The S3FW geo-domain ontology is considered as baseline ontology for all application oriented ontologies in order to provide shared common vocabulary for applications. Application ontology and ontology of user preferences are derived from this baseline ontology and merged in order to form specific application ontology.

3.2.1. Data Ontologies

In a GIS domain data ontologies are needed for mapping from relational spatial geo-databases to geo-domain ontologies. Data ontologies represent the underlying data model of a particular data source, for example a database of a specific geospatial product. Using data ontologies makes it easier to integrate different data sources used by LBS to domain ontologies and overcome heterogeneity of these sources. For our reverse geo-coding LBS application, we did not create data ontology but mapped geo-database directly to the domain ontology as we used specific geo-database from one specific provider Regio Ltd. The need for data ontology is growing when trying to solve the heterogeneity problem in more general case.

3.2.2. Geo-spatial Ontologies

There is a broad range of geo-spatial ontologies defined by several communities for different applications. Most widely used ontologies are GeoRSS Simple [14] that is designed for representing basic geometries (e.g. point, line, box, and polygon) and GeoRSS GML [14] that is created for representing more complex geometries and is standardised.

After the evaluation of different ontologies, we decided to reuse one of the existing geo-spatial ontologies namely GeoConcepts ontology developed in the FP6 CINESPACE project [13]. It is based on GeoOWL [14] standard and Geonames² feature type hierarchy. The GeoConcepts ontology expresses minimum number of geo-feature types and spatial relationships among them. It also makes possible to represent simple geometry of features as well as their location (position). This ontology is easily extendable, what was one of the reasons for our choice. Another reason was that we liked to build our approach as much as possible on standards in the geospatial field.

3.2.3. S3FW Geo-domain Ontologies

The baseline geo-domain ontology is derived from GeoConcepts ontology. Application and user preferences ontologies in turn are derived from this baseline ontology.

3.2.3.1. Taxonomy of Spatial Features

Figure 3 shows a fragment of domain ontology presenting only the particular subclasses that are used in examples outlined in this paper. By now the ontology contains 60 feature types in total and it will be extended as usage of this ontology in applications grows.

² www.geonames.org

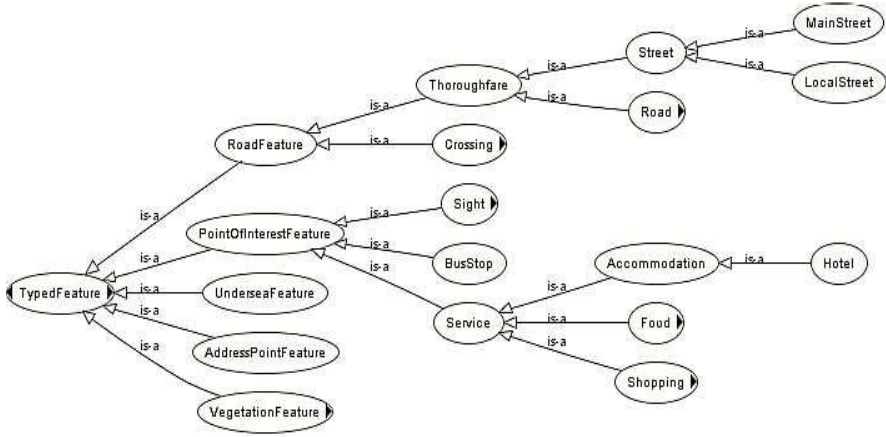


Figure 3. The baseline geo-domain ontology for S3FW (screenshot from the Protégé editor)

The first level subclasses of *TypedFeature* class are reused from GeoConcepts ontology, all other subclasses are originally created for S3FW domain ontology.

In order to be able to capture relevance of a spatial feature to the user, spatial features are enriched with ranking characteristic representing relevance. This is done via data-type property *featurePopularityMeasure* that takes values from 0 to 1. The values are given by GIS application for each instance of a feature type used for LBS.

Ontological model that would meet requirements of S3FW should enable the qualitative reasoning over spatial relationships that should be also expressed in ontology.

3.2.3.2. Spatial Relationships in Domain Ontology

We consider two types of spatial relationships among geospatial features: topological and semantic spatial relationships.

Topological spatial relationships are usually supported by various spatial operators of GIS engines or database systems like Oracle Spatial [15]. There is also an effort to create spatial ontologies that would model these qualitative relationships like RCC [16, 17]. The RCC is an axiomatization of certain spatial concepts and relations in first order logic. RCC8 [17] and SQL spatial operators define very specific relationships between geometries as for example Part of, Equal to, Overlaps, Proper Part of, etc. As a rule, these usually do not correspond to the *semantic* spatial relationships that domain experts (or end users) would expect.

Ontologies can help to overcome this limitation by explicitly defining the meaning of certain spatial relationships. This could be particularly useful for rather vague spatial relationships like “near to”, “next to”, “in between”, etc. We call these spatial relationships that capture the user expectations *semantic spatial relationships*. These are personalized semantic spatial relationships that are enriched with meaning that users will expect.

For the S3FW domain ontology, we reused topological spatial relationships (e.g. *hasSpatialPartOf*, *isPartOf*, etc.) from GeoConcepts ontology where they are represented using corresponding object properties with domain and range of some typed feature.

For representing original *semantic* spatial relationships we created class called *SpatialRelation*. Each specific semantic spatial relationship is defined as the corresponding subclass of *SpatialRelation* class.

For example, for our use case, semantic spatial relationships “is near to” and “is between” have been defined by the classes *Near* and *Between* correspondingly. As a rule, these relationships are n-ary relationships because the mentioned spatial relationships are either themselves n-ary (e.g. “in between”) or they have a property that measures their strength. The value of this characteristic is between 0 and 1 and it is modelled by the datatype property *strengthOfSpatialRelationship*. Its value is calculated by GIS application program.

For example, the class *Near* describes semantics of spatial relationship “is near to” by the following necessary condition presented in class expression language of the Protégé ontology editor:

```
SpatialRelation
and strengthOfSpatialRelationship exactly 1 Literal
and isNear exactly 2 TypedFeature
```

where *isNear* is an object property that expresses corresponding relationship between the instances of the classes *Near* and *TypedFeature*.

In principle, it is possible to define more *semantic* spatial relationships following our approach, for example, one can make use of *semantic* spatial relationships like “above”, “on corner of” “opposite”, etc. The approach is highly flexible and extendable; one can define *semantics* of any spatial relationship as needed for an application and take into account preferences of a certain user group.

4. Personalization of LBS

4.1. User Profiling

A user profile is needed in order to personalize the information to be delivered to the mobile device by LBS. In general, the user profile consists of personal data about the user, the user’s individual interests and preferences. Interests can be considered rather static and as such they can be defined in advance for LBS. Preferences may depend on the situation the user is in and on external factors. As situations in turn depend on context, then besides the user profiles context-specific demands of the user should be taken into account in personalization of LBSs. Typical context information for LBS is the user’s location and the actual time. Comparing to user profiles, contexts are dynamic. They are extracted from sensors in order to identify in which situation a user might be at a certain time. The user profile can be defined explicitly by the user and/or it can be learned based on the user’s previous behavior.

There are many works on how to express user profiles by ontologies [2, 3, 4]. Recent recommendation by W3C is the specification of a formal vocabulary RECO³ for recommendations in the Semantic Web. It provides general ontology in OWL DL for formalizing concepts and properties in order to represent different kind of user preferences and demands. The central concept of the ontology is preference, which is

³ <http://ontologies.ezweb.morfeo-project.org/reco/spec>

defined as a desire of an agent about properties of objects. This is for allowing to describe users' interests in RDF/OWL language. An interesting feature of the concept preference is that it allows ranking preferences according to the numeric value of the property *reco:utilityvalue* that determines the salience of the preference for the user.

As main goal in the RGCS case is to retrieve objects relevant to the user and being in relevant spatial relationships with the user location then it will be interesting in the future to use RECO ontology (at least partially) for user profiling in our framework. There is certain similarity with RECO approach and S3FW approach as both uses a property to express utility of some candidate object in satisfaction of the user demands. In S3FW case, utility of certain feature types (e.g. Hotels, Shops, and Crossings) is expressed via datatype property *featurePopularityMeasure* that ranks popularity of the spatial feature for some predefined user groups. However, in S3FW we do not yet express the user profiles for each individual user by ontologies. Personal information about the user and the user's interests are collected by application and used by application for deciding what user preferences class from ontology is to be used for filtering features.

In S3FW, another property *strengthOfSpatialRelationship* is used for ranking strength of spatial relationships. This property is related to the current location of the user (obtained by application) and its spatial relationship with some spatial feature. Bigger is the value of the property stronger is the relationship. That kind of ranking allows filtering out weak spatial relationships that are possible not interesting for the user.

In S3FW, we merged the application and user preferences ontologies in order to provide a more efficient service. For example, according to the user profile our application can make classification of instances that would best match the user preferences in certain feature types and at the same time take into account application specific requirements for classification.

By now, classes of user preferences are defined by application developers. In principle, we are in process to extend the system by the user profile ontology as well as to connect it to an application to automatically collect the user profile characteristics and generate user preferences classes.

Currently, the user preferences ontology consists of 2 classes (see Figure 4) and their subclasses that define particular restrictions for certain preferences of certain user groups.



Figure 4. An example of the user preferences ontology (screenshot from the Protégé editor)

Subclasses of the class *FeaturePreferences* define the user expectations about relevant spatial features. On the other hand, subclasses of the class *SpatialRelationPreferences* define the user expectations about spatial relationships. As

a rule, definitions of preferences on spatial relations are built on the definitions of preferences on feature types.

For example, for our current RGCS and its user groups we defined several classes of user preferences as subclasses of *FeaturePreferences* class. One example is the class *PopularHotelsOnGrossings* defined by using the following necessary and sufficient condition:

```
DescribedLocation
or (Hotel
    and featurePopularityMeasure some double[>= 0.7])
or (CrossingMainStreet
    and CrossingSameStreet)
```

This definition means that highly relevant spatial features for the user are hotels having popularity higher than 0.7 or crossings formed by main streets and the street where the described location lies. This definition again is used for defining relevant spatial relations for spatial reasoning as follows.

4.2. Feature Filtering based on User Preferences

The knowledge base for reasoning consists of the ontologies as defined above and of the actual class instances. Currently, there are several DL inference engines available for reasoning over OWL ontologies, e.g. Fact++ [18], RacerPro [19], and Pellet [20]. A DL system offers services to reason about the content of a knowledge base by providing standard reasoning services such as satisfiability, subsumption, and instance checking [21]. We used Pellet 1.5 as DL reasoner as it works efficiently with datatype properties and provides effective Abox reasoning mechanism. We used Protégé ontology editor [22] and Jena ontology API [23] for ontology management and access.

In this work, we combined quantitative and qualitative approaches for geospatial reasoning. First of all, quantitative reasoning was used in order to decide about the existence of a certain qualitative semantic spatial relationship (e.g. “is near to”). After ontology was populated with candidate entities participating in these relationships, qualitative reasoning was used for classification of individuals into predefined classes defining different user preferences and other application restrictions.

For example, in order to specify a set of spatial features that are near to the user location and highly relevant according to the user preferences the class *NearToPopularHotels* is defined as follows:

```
Near
and isNear some PopularHotelsOnGrossings
```

If we would like to take into account strength of the spatial relationship, then we may define the class *VeryNearToPopularHotels* based on the previous definition as follows:

```
NearToPopularHotels
and strengthOfSpatialRelationship some double[>0.7]
```

The first definition means that at least one of the two instances in the relevant to the user near relation must be of the class *PopularHotelsOnGrossings* (cf. definition in section 4.1), i.e. a feature that is highly relevant according to the user preferences. The second definition as a subclass of the previous class makes a set of instances satisfying the definition more specific according to the value of the property *strengthOfSpatialRelationship*.

We now may like to retrieve the instances of these concepts. Reasoning is required in order to obtain these instances since there are no given instances of *NearToPopularHotels* or *VeryNearToPopularHotels* classes.

Automatic classification is used to filter out geospatial features that are insignificant according to the user profile and application context. Therefore, the defined classes in ontology of user preferences can be considered as filters or instance retrieval queries.

The RGCS use case demands very limited output consisting as maximum of 1-4 instances. However, the result of automatic classification may contain more relevant features and spatial relationships to recommend to the user. To overcome this problem, we may rank resulting set of spatial relationships according to their strength and related spatial features according to their value of *featurePopularityMeasure*. This procedure gives more precise retrieval output. Another solution is that for current application, we may pick up just one individual from each different class of retrieved instances. This is because in order to describe the user location it is not needed, for example to recommend all the nearby popular hotels as one may be enough to give the user an idea, where the user is located.

5. Verbalization of Filtered Candidate Objects

The outcome of ontology-based retrieval of salient objects and their relationships with the user location is a collection of spatial objects and relationships, encoded in technical, machine oriented language RDF/XML [24]. This format needs to be converted into human-readable, natural language alike textual representation – a task of natural language generation (NLG) called verbalization.

With this task we aim no absolutely rigid grammar, but try to reach a clearly understandable, readable and concise text. We are using anaphoric references to avoid repetitions. The outcome of verbalization process may be in turn an input to speech synthesizer.

From the usability point of view, we demand an easy way to translate and customize the verbalizer into new languages, preferably at a level, which could be carried out by non-technical persons – linguists with no need to re-program the code.

5.1. Methodology Used for Verbalization

We use Adaptive Natural Language Generation (ANLG). ANLG is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information [5]. In our S3FW architecture we translate semantic structures, presented in RDF, into human readable text.

For this verbalizer-component we use a pipeline pattern from [5] as follows:

- Content determination, text planning, discourse structuring;

- Sentence planning for selecting attributes for referring expressions, aggregating content into sentence-size units, and selecting lexical items,
- Surface realization that converts sentence plans into natural language.

We also considered the using ACE verbalizer [25], OMG HUTN [26] and other out-of-the-box solutions, but all of them turned out not to satisfy the requirements of our use case as we expected multilingual verbalization.

We utilized template-based NLG approach that maps nonlinguistic input *directly* (i.e., without intermediate representations) to the linguistic surface structure [5]. Usually, templates are considered most suitable for shallow forms of generation, in which the templates contain predefined surface strings. Templates also are not always adaptable for new applications and need to be recreated. However, the approach met our requirements so far as we are rather at initial stage with NLG process. It might happen that we need deeper forms of generation.

5.2. An Example of Verbalization

The starting point for the text generation pipeline is a set of salient objects with their relevant spatial relationships represented in RDF. A fragment of this representation is as follows:

```
...
<!-- file:/c:/try/regio_geoonto.owl#Theatre -->
  <owl:Class rdf:about="#Theatre">
    <rdfs:label xml:lang="en">theatre</rdfs:label>
    <rdfs:label xml:lang="et">teater</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Sight"/>
  </owl:Class>
  <regio_geoonto:Theatre
rdf:about="&regio_geoont_1;genid-A37">
    <rdfs:label>Vanemuine</rdfs:label>
    <regio_geoonto:featurePopularityMeasure
rdf:datatype="&xsd;double">1.0</regio_geoonto:featurePopul
arityMeasure>
    <regio_geoonto:id
rdf:datatype="&xsd;int">6</regio_geoonto:id>
  </regio_geoonto:Theatre>
  <!-- file:/c:/try/regio_geoonto_relations.owl#near1 -->
    <Near rdf:about="#near1">
      <regio_geoonto:strengthOfSpatialRelationship
rdf:datatype="&xsd;double">1</regio_geoonto:strengthOfSpat
ialRelationship>
      <isNear rdf:resource="&regio_geoont_1;genid-A37"/>
    </Near>
  </rdf:RDF>
```

We used *rdfs:label* property to specify that for example the class *Theatre* is typically expressed in natural language by words such as “*Theatre*” (in English) or “*Teater*” (in Estonian), etc. (see example above). In the example, the instance *genid-A37* is labeled as “*Vanemuine*”.

Let's take the spatial relationship "near" from our ontology (cf. section 4). The "near" relationship should be used by the verbalizer as in a template like

```
YOUARE (language) + TERM(NEAR, language) +  
LABEL (FACT1_NAME, language) +  
TERM (FACT1_TYPE, language)
```

For example in English the output text corresponding to RDF input and the template is as follows: "You are near Vanemuine Theatre".

6. Evaluation of the Approach

We have tested the S3FW with a technically challenging use case, namely by personalized reverse geo-coding service (see chapter 1). The use case controller handler was used to deal with all system events of the use case. The main purpose of the use case handler was to coordinate the tasks between a proprietary GIS solution and the S3FW's services in order to guarantee selecting of a preliminary set of candidate features and populating ontology with them, filtering candidate features according to user preferences, and verbalizing the results to a natural-language-alike text.

The ontology has been implemented in the OWL DL [7] by using the Protégé ontology editor [22]. Its Description Logic expressivity is SHIQ (D).

For the RGCS use case implementation, we populated our domain ontology with instances by using the Jena Ontology API [23] and an application program written in Java that performed the needed queries to geo-databases provided by Regio Ltd. The application program also calculated for each instance of a geo-feature type (subclass of *TypedFeature* class) the value of datatype property *featurePopularityMeasure* and for each instance of defined spatial relationship the value of datatype property *strengthOfSpatialRelationship*. For ontology population, a set of candidate features for relevant instances was selected from all available features that were spatially related to the given location. Usually, there are hundreds of candidate relevant features from what the most relevant ones (usually 1-4) will serve for the description of the location to the user in natural language. The DL reasoner Pellet [20] was used for filtering out insufficient features and spatial relationships.

7. Related Works

Most of related works fall into two categories: creating geospatial ontologies and providing architectural solutions that support ontology based retrieval and reasoning over spatial relationships. In addition, works on ontology verbalization are important for user interface implementation of LBS.

A lot of effort is devoted to building geospatial ontologies [28, 29, 30] as already discussed in section 3. There are also proposals for automatically extracting geospatial ontologies from geo-databases [12, 31]. It is shown in [12] that adding a semantic layer to a database makes it possible to refer to concepts that have no direct correspondence to the database table. This gives a semantic view to geographical data. Some researchers propose to use conceptual models of databases as data ontologies [31]. As

mentioned above, in this paper we do not develop data ontologies, nevertheless, we appreciate work what is done in this field.

Architectural solutions that support geo-spatial ontologies and spatial reasoning fall into the following directions:

- DL extensions based on the definitions of a concrete domain e.g. spatial domain [32]. However, their approach needs to be extended to be practical as only geometry class, polygon, is defined together with a set of predicates representing topological relationships between two spatial objects.
- Hybrid approaches that combine different paradigms e.g. DL and GIS databases. In [33], a hybrid deductive GIS with DL-component is proposed. Using this approach, the DLMAPS system [34] provides spatio-thematic query-answering in the domain of city maps. Query expansion with ontology concepts is considered in the QuONTO system [35] but in this system no ABox retrieval was needed as database systems are used instead.
- Pragmatic approaches. For example, in [36] geo-ontologies are built on OGC-GML standard and provide extensions to OWL-S for service profiles. In [37] a method is presented that enables ontology based query of spatial data available from Web Feature Services (WFS) and from data stored in databases. In contrast to our OWL ontologies, their method uses RDF ontology, RDF views to database tables and SPARQL as a query language.

Comparing to related approaches listed above, an architectural solution proposed in our work can be considered as a pragmatic hybrid approach. We combine already existing technology and standards as we use OWL DL as ontology language and Pellet as DL reasoner, we reuse and extend GeoConcepts ontology for geo-features and GIS database for obtaining facts. However, our approach is different of the any above mentioned approaches in the following: we merge geospatial, the user profile and application ontologies in order to perform ontology based retrieval of spatially relevant features and relations. For this purpose, we populate ontology on the basis of GIS databases and use Abox reasoning to find suitable features. After that OWL DL descriptions of retrieved features are automatically converted to human readable language.

With respect to ontology verbalization efforts, a number of works on NLG are devoted to general methodology of NLG [5, 38], but less work is done on generating natural language alike texts on the basis of RDF or OWL [39].

8. Conclusion

In this paper we presented ontology based solution to the personalization problem of LBSs. The provided ontology-driven development approach for personalized LBS uses ontologies for explicitly representing geo-spatial domain concepts and user preferences as well as for the multilingual verbalization of the results of feature filtering.

Novelty of our contribution is twofold; it makes LBS development independent on heterogeneity of data sources and enables personalization of LBS. Personalization is based by enriching purely topological spatial relationships with meaning that end users will expect and reasoning on such *semantic* spatial relationships.

As a proof of the concept and evaluation of the proposed approach, we developed a personalized reverse geo-coding service where traditional reverse geo-coding is

extended by providing multilingual textual description of relevant to the user spatial features that are in personalized spatial relationships with the actual location of the user.

Acknowledgments

This research was partially supported by European Regional Development Fund within the framework of the project “Smart ontology-based spatial information retrieval and integration” of ELIKO Competence Centre in Electronics-, Info- and Communication Technologies and by the target-financed theme No. 0322709s06 of the Estonian Ministry of Education and Research as well as by the ERDF funded Estonian Centre of Excellence in Computer Science, EXCS.

References

- [1] Virrantaus, K., Markkula, J., Garmash, A., Terziyan, Y.V.: Developing GIS-Supported Location-Based Services. In: First International Workshop on Web Geographical Information Systems, pp. 423-432. Japan (2001)
- [2] Middleton, S., De Roure, D., Shadbolt, N.: Capturing Knowledge of User Preferences: Ontologies in Recommender Systems. In: First International Conference on Knowledge Capture, pp. 100-107. ACM Press (2001)
- [3] Yu, Y., et al.: Recommendation systems using location-based ontology on wireless internet. An example of collective intelligence by using “mushup” applications, *Expert Systems with Applications*. 36, 11675-11681 (2009)
- [4] Robal, T., Haav, H.-M., Kalja, A.: Making Web Users Domain Models Explicit by Applying Ontologies. In: Hainaut, J.-L., et al. (eds.) *ER Workshops 2007*. LNCS vol. 4802, pp. 170-179. Springer, Heidelberg (2007)
- [5] Reiter, E., Dale, R.: *Building Natural Language Generation Systems*. Cambridge University Press (2000)
- [6] Google Maps API Reference, code.google.com/apis/maps/documentation/reference.html
- [7] OWL DL, <http://www.w3.org/TR/owl-guide>
- [8] Haav, H.-M., Kaljuvee, A., Luts, M., Vajakas, T.: Ontology-based retrieval of spatially related objects for location-based services. In: Meersman, M., Dillon, T., Herrero, P. (eds.) *OTM 2009*, LNCS vol. 5871, pp. 1010-1024. Springer, Heidelberg (2009)
- [9] Fonseca, F.T., Egenhofer, M.J., and Agouris, P., Using Ontologies for Integrated Geographic Information Systems, *Transactions in GIS*. 6, 231-257 (2002)
- [10] Zipf, A.: User-Adaptive Maps for Location-Based Services (LBS) for Tourism. In: Woeber, K., Frew, A., Hitz, M. (eds.) *9th International Conference for Information and Communication Technologies in Tourism*, Springer Computer Science, pp. 329-338, Springer-Verlag, Wien (2002)
- [11] Cheng, G., Du, Q., et al: Research on Designing Ontologies for Location-based Services. In: Gong P., Liu Y. (eds.) *Geoinformatics 2007, Proceedings of SPIE*, vol. 6754 (2), pp. 67542X.1.1-67542X.11. SPIE, Bellingham, Washington (2007)
- [12] Tryfona, N., Pfoser, D.: Data Semantics in Location-Based Services. In: Spaccapietra, S., Zimányi, E. (eds.) *Journal on Data Semantics III*. LNCS vol. 3534, pp. 168-195. Springer, Heidelberg (2005)
- [13] CInSPACE project, <http://www.cinespace.eu>
- [14] GEO RSS, <http://www.georss.org/>
- [15] Oracle Spatial, <http://www.oracle.com/us/products/database/options/spatial/index.htm>
- [16] Randell, D.A., Cui, Z., Cohn, A. G.: A spatial logic based on regions and connections. In: *3rd International Conference on Knowledge Representation and Reasoning*, pp. 165-176. Morgan Kaufman (1992)
- [17] Grütter, R., Scharrenbach, T., Bauer-Messmer, B.: Improving an RCC-Derived Geospatial Approximation by OWL Axioms. In: Sheth, A. et al. (eds.) *ISWC 2008*, LNCS vol. 5318, pp. 293–306. Springer, Heidelberg (2008)
- [18] Fact++, <http://owl.man.ac.uk/factplusplus>
- [19] RacerPro, <http://www.racer-systems.com>

- [20] Pellet, <http://www.clarkparsia.com/pellet>
- [21] Baader, F. et al. (eds.) *The Description Logic Handbook*, Cambridge University Press (2003)
- [22] Protégé Ontology Editor, <http://protege.stanford.edu>
- [23] Jena Ontology API, <http://jena.sourceforge.net/ontology/index.html>
- [24] RDF standard, <http://w3.org/TR>
- [25] ACE verbalizer, http://attempto.ifi.uzh.ch/site/docs/owl_to_ace.html
- [26] OMG HUTN, <http://www.omg.org/technology/documents/formal/hutn.htm>
- [27] Larman, C.: *Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall PTR (2005)
- [28] Ressler, J., Dean, M., Kolas, D.: *Geospatial Ontology Trade Study*, National Geospatial-Intelligence Agency, http://projects.semwebcentral.org/docman/?group_id=84
- [29] W3C geo ontology, <http://www.w3.org/2003/01/geo/#vocabulary>
- [30] W3 GeoIncubator Group, <http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023>
- [31] Baglioni, M., Masserotti, M. V., Renso, C., Spinsanti, L.: *Building Geospatial Ontologies from Geographical Databases*. In: Fonseca, F., Rodríguez, M.A., Levashkin, S. (eds.) *GeoS 2007*, LNCS vol. 4853, pp. 195–209. Springer, Heidelberg (2007)
- [32] Haarslev, V., Lutz, C., Möller, R.: *Foundations of spatioterminological reasoning with description logics*. In: Cohn, A.G., et al. (eds.) *6th International Conference on Principles of Knowledge Representation and Reasoning*. pp. 112–123, Trento (1998)
- [33] Wessel, M.: *Some Practical Issues in Building a Hybrid Deductive Geographic Information System with a DL Component*. In: *10th International Workshop on Knowledge Representation meets Databases*. CEUR-WP, vol. 79, pp. 99–111. (2003)
- [34] Wessel, M., Möller, R.: *Flexible Software Architectures for Ontology-Based Information Systems*. *J. of Applied Logic. Special Issue on Empirically Successful Systems*. 7, 75–99 (2009)
- [35] QuONTO project, <http://www.dis.uniroma1.it/~quonto/>
- [36] Shen, Y., et al: *A Pragmatic GIS-Oriented Ontology for Location Based Services*. In: *19th Australian Software Engineering Conference*, pp. 562–569. IEEE Press (2008)
- [37] Zhao, T., Zhang, C., Wei, M, Peng, Z-R.: *Ontology-Based Geospatial Data Query and Integration* In: Cova, T.J. et al. (eds.) *GIScience 2008*, LNCS vol. 5266, pp. 370–392. Springer, Heidelberg (2008)
- [38] Dale, R., Geldof, S., Prost, J-P.: *Using Natural Language Generation in Automatic Route Description*. *J. of Research and Practice in Information Technology*. 37, 89–105 (2005)
- [39] Wilcock, G.: *Talking OWLs: Towards an Ontology Verbalizer*. In: *2nd International Semantics Web Conference. Human Language Technology for the Semantic Web and Web Services*, pp. 109–112. (2003)

Interoperability Services for Models and Ontologies¹

Jürgen EBERT, Tobias WALTER

University of Koblenz-Landau

Universitätsstr. 1

D-56070 Koblenz

e-mail: {ebert,walter}@uni-koblenz.de

Abstract. Model-based approaches in the UML/MOF technological space and ontology-based approaches in the OWL technological space both support conceptual modeling using different kinds of representation and analysis technologies. Both spaces provide services for accessing and manipulating models and ontologies, respectively.

This paper compares both spaces using to a mapping of models and ontologies based on a common model-theoretic semantics. Based on this mapping, different services for bridging the respective technological spaces are defined and discussed. Three different bridging strategies, namely Adaptation, transformation, and integration, are distinguished.

Introduction

Models are central artifacts in model-driven software engineering. They are created, edited, and transformed during software development, and querying and reasoning is performed on them. The concrete languages, techniques, and tools used to support these activities depend on the environment the software development processes is performed in. Modeling in general can be done in different *technological spaces* [1], e.g., using UML-like modeling in an eclipse-based environment or using ontology-based modeling making extensive use of reasoning facilities.

The heterogeneity of and the insufficient interoperability between technological spaces often hinders the simultaneous use of their respective advantages. In this paper, which is an enhanced version of [2], we tackle the interoperability problem between the UML-inspired world which we call *Modelware* and the ontology-inspired world, called *Ontoware*. We investigate different ways of bridging between those two spaces.

To define these bridges between the spaces, their similarities (commonalities) and their dissimilarities (variabilities) are identified first. These are elaborated by specifying the semantics of the modeling languages of both worlds in a common approach (namely graph-based model-theoretic semantics), which leads to a mapping of the underlying concepts of both. This mapping is used as a basis for building concrete bridges. We

¹This work has been partially funded by the EU (grant number 216691 in the 7th framework programme), see <http://www.most-project.eu>.

discuss the three main bridging approaches and derive a set of services needed to support the respective kinds of interoperability.

Modelware and Ontoware. Models are used to describe those parts of reality that are relevant for a software system to be built (domain models, *descriptive modeling*), and models are also used to prescribe the target system (metamodels, *prescriptive modeling*). Descriptive modeling is also called conceptual modeling in the knowledge representation literature. Conceptual models in particular describe the relevant classes of entities and relationships in the domain of interest. We use the term *conceptual model* as a general term, whereas the term *model* applies to concrete artifacts from the Modelware world and *ontology* pertains to artifacts from the Ontoware world.

With the advent of the Semantic Web, ontologies found their way into software development, as well [3,4]. Though in the stronger sense an ontology is “a formal explicit specification of a shared conceptualization for a domain of interest” [5], the term ontology is today actually being used for logical knowledge bases in a broader sense. Since Semantic Web technology is heavily based on description logics (DL) [6] in particular, the term seems even more and more to be used as a synonym for description-logics-based knowledge bases in general. This DL-based way of modeling is shortly characterized as *Ontoware* in the following, opposed to MOF-like modeling using *Modelware*, where structural models are seen as networks of objects which can easily be implemented in modern object-oriented languages.

Goal of this paper. Common use of the two technological spaces of Modelware and Ontoware and bridging them in particular requires a thorough understanding of the similarities and dissimilarities of both spaces. Thus, to investigate more deeply on this topic in order to really “marry” both worlds, *concrete languages* with concrete syntaxes and semantics have to be considered. On the Modelware side there is a predominance of OMG’s UML/MOF-based approaches including Ecore as Eclipse’s variant, whereas on the Ontoware side the W3C description languages of the OWL family seem to be the most important.

In this paper, we use *UML and OCL* [7] as representatives for Modelware and *description logics (DL)* [8] as the respective representation for Ontoware². We shortly introduce both approaches, show that they can be mapped to by graph-based semantics, and discuss the three kinds of bridging based on this mapping.

The paper is structured as follows. Sections 1 and 2, respectively, give a short introduction to both technological spaces, explaining their main characteristics and sketching their formal bases. Section 3 discusses how both worlds can be mapped, and Section 4 explicates and discusses different ways of bridging them. Section 6 concludes the paper.

1. Modelware

Modeling has a long history in database and software engineering. A long chain of seminal work on modeling starting with Chen’s ERM [10], over several variants like NIAM [11], James Martin’s approach [12], the books of the Booch [13], Rumbaugh et al. [14], Jacobson [15] and many others lead to the current de-facto standardization given by

²Description logics are the pure logical formalisms behind ontology languages, like OWL 2 [9]. Since DL ontologies are much more compound than OWL ontologies, they are used here for brevity’s sake.

OMG's *Unified Modeling Language* (UML) [7], whose class and object diagrams offer means to notate conceptual models. Though UML [7] is quite helpful as a “unifying” approach, there also are other, richer technologies in Modelware (e.g. ADOxx [16], Ker-meta [17], and the TGraph Approach [18]) that have more expressive modeling elements.

Much work on software development (including model driven development [19] and model transformation [20]) and much implementation work (including several UML tools and the Eclipse³ project) build upon UML, mostly driven by the Object Management Group (OMG)⁴. Thus, a rich and well understood “technological space” has developed which is widely accepted and gets steadily extended by the Modelware community.

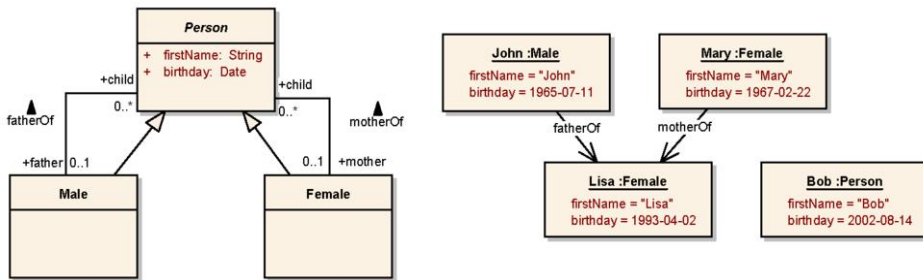


Figure 1. Two UML Diagrams

Example. To underpin the messages of this paper the often used and easy-to-understand family example is used. Figure 1 depicts a model of a genealogy as a *UML class diagram*. The description consists of classes (*Person*, *Male*, *Female*), which represent sets of objects, and associations (*fatherOf*, *motherOf*), which represent sets of links between these objects. Both, classes and associations, are classifiers, i.e. they have a name and stand for sets of instances. Classes may have named attributes with a respective domain, and they may generalize others (*Person* is a generalization of *Male* and *Female*). A corresponding set of instances is described by a *UML object diagram* in Figure 1. Here, the classes of the objects and the associations of the links are added as their types, and values are included for the type-specific attributes.

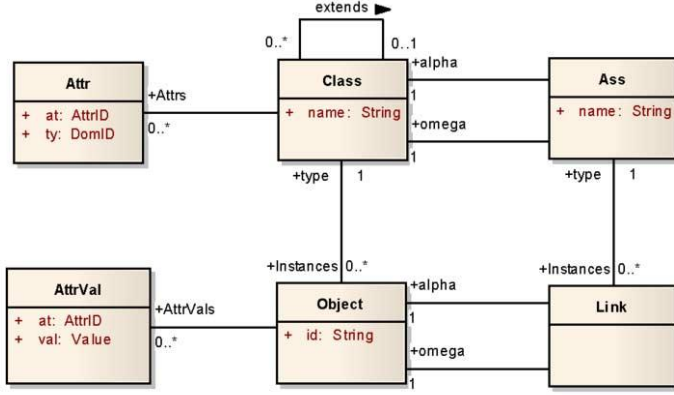
To define further restrictions for the sets of instances, additional constraints may be added. Class diagrams support the notation of *multiplicities* for associations which have been added beside the roles in Figure 1. Note that ‘0..1’ (instead of ‘1’) has been added to the parent roles, since a *closed world* is assumed by UML, which implies that all instance sets are finite and therefore there must be *Persons* whose *father* or *mother* is not in the instance set.

OMG proposes to use *OCL* [21] for the definition of *further constraints*. E.g., to describe that parents are born before their children, the following constraint might be added:

```
context Person
inv: father.birthday < self.birthday
    and mother.birthday < self.birthday
```

³<http://www.eclipse.org>

⁴<http://www.omg.org>



Let $Class$ be the set of classes and Ass be the set of associations in a class diagram CD where $\alpha(a)$ and $\omega(a)$ denote the start and end class for each association a .

Furthermore, let $AttrID$ and $TypeID$ be the universes of attribute and type identifiers, respectively, and let $Attrs(c) = \{(at_1, ty_1), \dots, (at_{k_c}, ty_{k_c})\}$ be the set of $AttrID$ - $TypeID$ -pairs for each class $c \in Class$ (under inheritance). $extends : Class \leftrightarrow Class$ is the relation defining the specialization between a subclass and its superclass

Then, the semantics of CD is the set of interpretation functions I , which define the set of instance graphs G^I for CD . An interpretation function I assigns disjoint sets of vertices to the classes, disjoint sets of edges to the associations and some subset of the universe $Value$ to the elements of $TypeID$.

An instance graph $G^I = (V, E, \alpha, \omega, attr)$ with $\alpha : E \rightarrow V, \omega : E \rightarrow V$, $attr : AttrID \times V \rightarrow Value$ has to fulfill the following conditions:

- $V = \bigcup_{c \in Class} c^I$ is the vertex set of G^I
- $E = \bigcup_{a \in Ass} a^I$ is the edge set of G^I
- $\forall e \in E : t(\alpha(e)) = t_s \wedge t(\omega(e)) = t_t$, with $t_s extends^* \alpha(t(e))$ and $t_t extends^* \omega(t(e))$
- $\forall v \in V, (at, ty) \in Attrs(t), t(v) extends^* t : attr(at, v) \in ty^I$

Here, we use the function $t(x) = d \Leftrightarrow x \in d^I$ for $d \in (Class \cup Ass)$ to express that x has type d . $extends^*$ describes the transitive specialization relation between two types.

Let $Object$ be the set of objects and $Link$ be the set of links in an object diagram OD where $\alpha(l)$ and $\omega(l)$ denote the start and end object for each link l .

Then, we have an instance graph G^I with

- $V = \bigcup_{c \in Class} \{o^I \mid o \in Object \wedge type(o) = c\}$
- $E = \bigcup_{a \in Ass} \{l^I \mid a \in Link \wedge type(l) = a\}$
- $\forall l^I \in E : \alpha(l^I) = (a(l))^I \wedge \omega(l^I) = (\omega(l))^I$
- $\forall o^I \in V, (at, x) \in AttrVals(o) : attr(at, o^I) = x$

Figure 2. Graph Semantics for Models (Simplified Excerpt)

Formal Foundation. Starting with Chen's [10] work, the formal semantics of models has mainly been defined using constructive set theory and predicate logics, leading to a clear *model-theoretic semantics*: For a given diagram D its extension, i.e. the set of its instances, is described.

Graphs are a means for talking about semantics which is widely used in Modelware. A concise description of semantics can be given using *graphs* as instances [22]. With the advent of MDA graph-based views on modeling semantics have become even more important, since graphs form an easily understandable, mathematically sound and efficiently implementable basis for storing and manipulating model instances. Many Modelware tools are based on graphs, especially when model transformations are concerned (e.g., AGG [23], PROGRES [24], MOLA [25], GReTL [26]).

A model-theoretic semantics for models defines the set of legal instances that are described by a model. In Figure 1 one instance graph of the genealogy model is sketched by the object diagram in the same figure.

Instance graphs have objects of the respective diagram classes as *vertices* and links corresponding to the respective diagram associations as *edges*. The edges are assumed to be directed according to their reading direction as expressed by the small black triangles at their association. The classes and associations serve as *types* of the instance vertices and edges. The types of the edges and their respective endpoints are compatible with the incidences between associations and classes in the model. Vertices may carry attribute-value-pairs as defined by their type (and all its supertypes to include inheritance) in the model.

Based on such a formal, unambiguous semantics the set of models defined by a conceptual diagram can be restricted by additional logical formulae. Thus, the formulae of the constraint language lead to additional constraints on the graph instances. Using the model-theoretic semantics of Schmitt [27], OCL constraints can directly be checked on instance graphs. To enable checking of the instance graph, *constraint languages* usually are algorithmically feasible sublanguages of predicate logic, which allow for model checking by assessing the conformance of instance sets. This also holds for OCL.

Summarizing, Modelware separates structural modeling from (additional) constraints by using different description paradigms, namely object-relationship descriptions and predicate logics which together define the legal sets of instances. Though the usage of constraint languages in practice does not seem to be so wide-spread as pure class diagrams, in principle the Modelware modeling facilities can be characterized as a combined use of visual class models and additional constraints which can be described precisely using a model-theoretic semantics.

2. Ontoware

The use of logics to describe conceptual models has a long history in works on knowledge-based systems in Artificial Intelligence. With the advent of the Semantic Web special logics like F-logic [28] and description logic (DL) [8] have been adopted. The former may be characterized as a strict format for descriptions in predicate logic, thus being undecidable in general. The latter is a family of sublanguages of predicate logic most of which are decidable having varying degrees of complexity.

The family of *description logics* [6] may be structured along a set of features described by calligraphic letters \mathcal{ALC} plus some of $\{\mathcal{F}, \mathcal{S}, \mathcal{H}, \mathcal{R}, \mathcal{O}, \mathcal{I}, \mathcal{N}, \mathcal{Q}\}$ which may be used to define a concrete DL language by the concept constructors it supports. Different sets of features lead to different expressivity and different complexity properties.

OWL (*Web Ontology Language*) is a family of precisely defined description logics introduced by the World Wide Web Consortium (W3C⁵). OWL documents are called *ontologies* by W3C. OWL comes with a set of corresponding notations and tools. The current version OWL 2 corresponds to the *SRIOQ*-subset of description logics. OWL 2 as a web ontology language is primarily defined by its abstract syntax [9]. Mostly, OWL 2 ontologies are serialized using RDF [29] as XML-documents. There are also some concrete syntaxes (e.g., Manchester style, functional style) which are better accessible to humans than pure XML. There are several sublanguages of OWL, called profiles, namely OWL 2 EL, OWL 2 QL and OWL 2 RL [30], which differ in their complexity. The OWL ontology languages are based on description logics (DL) [8].

Example. In description logics knowledge is described in two parts. The terminology component (*T-Box*) describes the concepts and their roles, whereas the assertion component (*A-Box*) specifies the individuals and their properties. Given the \mathcal{ALC} variant of DL, the model depicted in Figure 1 can be described as a DL-text in Figure 3 by axioms and assertions.

```
// T-Box
Male  $\sqsubseteq$  Person  $\sqcap \forall \text{fatherOf}.\text{Person}$ 
Female  $\sqsubseteq$  Person  $\sqcap \forall \text{motherOf}.\text{Person}$ 
Person  $\sqsubseteq \forall \text{name}.\text{String} \sqcap \forall \text{birthday}.\text{Date}$ 

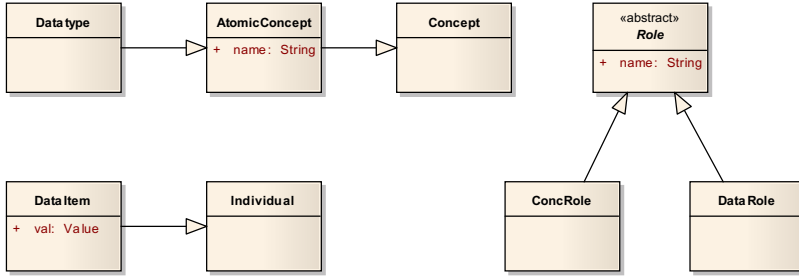
// A-Box
Person(John), Person(Mary), Person(Lisa), Person(Bob)
fatherOf(John, Lisa), motherOf(Mary, Lisa)
firstName(John, "John"), birthday(John, 1967-07-11)
firstName(Mary, "Mary"), birthday(Mary, 1967-02-22)
firstName(Lisa, "Lisa"), birthday(Lisa, 1993-04-02)
firstName(Bob, "Bob"), birthday(Bob, 2002-08-14)
```

Figure 3. A Description Logic Example

The first part of the description in Figure 3 is a T-Box which introduces the classes and their properties which are called *concepts* and *roles*, respectively, in pure DL. There are atomic concepts (Person, Male, and Female) and atomic properties (fatherOf, motherOf, name, and birthday). Further concepts are defined using the *constructors* $\forall r.c$ and $c \sqcap d$ for roles r and concepts c and d .

The *T-Box* consists of three *axioms*, each of which makes a statement about different (named or anonymous) concepts, e.g., it is stated that Males are Persons and all individuals they may be fatherOf are also Persons. Furthermore, it is stated that Persons may have a firstName, which is a String, and a birthday, which is a Date. Note, that relationships like fatherOf and attributes like name and birthday are not differentiated.

⁵<http://www.w3c.org>



Let *AtomicConcept* be the set of concepts and let *Role* be the set of roles in an *ALC* DL ontology *O*. Assuming *O* is schema-like, the roles are either concept roles (*ConcRole*) or datatype roles (*DataRole*).

Then, the semantics of *O* is the set of interpretation functions *I*, which define the set of instance Graphs G^I for *O*.

An instance graph $G^I = (V, E, \alpha, \omega, attr)$ with $\alpha : E \rightarrow V$ and $\omega : E \rightarrow V, attr : AttrID \times V \rightarrow Value$ has to fulfill the following conditions:

- $\forall c \in AtomicConcept : c^I \subseteq V$
- $\forall r \in ConcRole : r^I \subseteq E$
- $\forall s \in DataRole : s^I : \{s\} \times V \rightarrow Value$

Furthermore, there are additional conditions for the concept constructors in *ALC*

$\forall c, d \in Concept, r \in ConcRole, s \in DataRole$:

- $\top^I = V$
- $\perp^I = \emptyset$
- $(c \sqcap d)^I = c^I \cap d^I$
- $(c \sqcup d)^I = c^I \cup d^I$
- $(\neg c)^I = V \setminus c^I$
- $(\forall r.c)^I = \{v \in V \mid \forall w \in V, e \in r^I : v = \alpha(e) \wedge w = \omega(e) \Rightarrow w \in c^I\}$
- $(\exists r.c)^I = \{v \in V \mid \exists e \in r^I : \alpha(e) = v \wedge \omega(e) \in c^I\}$
- $(\exists s.c)^I = \{v \in V \mid \exists x \in Value : attr(s, v) = x\}$

According to the *axioms* in the ontology, the instance graph has to fulfill further conditions $\forall c, d \in Class$:

- $c \sqsubseteq d \Rightarrow c^I \subseteq d^I$
- $c \equiv d \Rightarrow c^I = d^I$

Let *Individual* be the set of individuals in *O*. Then we have $Individual^I \subseteq V$.

Finally, the *assertions* imply some constraints:

- $\forall (c, i) \in ConceptAssertion : i^I \in V$
- $\forall (r, i, j) \in RoleAssertion : \exists e \in r^I : \alpha(e) = i \wedge \omega(e) = j$

Figure 4. Graph Semantics for DL (*ALC*, Simplified Excerpt)

They both are treated equally by roles. The *A-Box* enumerates the individuals and their properties using concept and role assertions.

Formal Foundation. The meaning of ontologies is precisely defined in a comprehensive logic theory. Looking at them as models in the sense described above, a *model-theoretic semantics* seems to be the most adequate description for the purpose of bridging Ontoware and Modelware.

The model-theoretic semantics can also be given by *graphs*. Concepts (Person, Male, Female) correspond to vertex sets (which do not have to be disjoint), and roles stand for directed edges. Concepts are *not* viewed as types, but only as sets of vertices. Thus, a vertex can belong to several concepts. Consequently, subsumption (\sqsubseteq) corresponds to set inclusion and not to subtyping.

Figure 4 sketches a graph semantics, which is defined inductively over the set of concept constructors. Note, that in this case, the semantics describes only the inclusion of the interpretation sets in the graph. Since an *open world semantics* is assumed, there may be more vertices and edges in an instance graph than the enumerated individuals. In fact, the graph may even be infinite.

Roles correspond to attributes of the vertices, if their object is a basic value, and to edges otherwise. Assuming that **String** and **Date** are basic values, all graphs conforming to Figure 1 do also conform to Figure 3. The contrary is not necessarily true, since the open world assumption allows further vertices and edges in the instance graphs.

Whereas Modelware is based on explicit classes and explicit expression of *specialization*, Ontoware uses concepts as a general means for expressing properties. Thus, there are additional anonymous concepts and specialization (in the form of *subsumption*) is an implicit property.

3. Mapping

Conceptual models describe a domain in terms of its concepts and their relations and possibly also by sets of suitable instances and links. Both, UML class diagrams with constraints and object diagrams as well as description logic-based languages like OWL 2 offer means to describe conceptual models formally. Though both approaches stem from different technological space they have a lot of *commonalities*, as has been shown in Sections 1 and 2. Thus, there is some evidence, that bridging both worlds might be possible, i.e. that models and ontologies may be mapped to each other.

Any kind of bridging has to be based on a *mapping* of the notions of both worlds. The model-theoretic semantics sketched in Sections 1 and 2 show that the relevant parts of both worlds match quite well and give hints how such a mapping may be done. Assuming that the ontologies used as conceptual models in software development are *schema-like*, the graph-based semantics defined for Modelware and Ontoware, respectively, can be used as a basis for mapping. An approximate matching of notions derived from the semantics is visualized in Figure 5.

The graph-based semantic description in Sections 1 and 2 render a high similarity between both (simplified) modeling approaches, namely UML class diagrams, OCL and object diagrams on the one hand and description logical models (as representatives of ontologies) on the other hand. This similarity paves the way to defining a mapping between both worlds as a basis for a bridge.

UML	DL (schema-like)
Class	Atomic Concept
Association	Concept Role
Attribute	Datatype Role
Object	Individual
Domain	DataType
Subclassing	Subsumption

Figure 5. Equivalences between Models and Ontologies

- Classes in UML correspond to concepts in DL. They both stand for a specific set of vertices in the instance graph.
- The associations in UML correspond to concept roles in DL. They both stand for specific sets of edges in the instance graph.
- The attributes in UML correspond to datatype role in DL, since they associate objects to values from a predefined universe.
- The UML objects represent single vertices like the individuals in DL descriptions.
- The domains of attributes in UML comprise atomic values like the datatypes in DL.
- Subclassing in UML implies subsumption of the respective instance sets, but it furthermore implies inheritance, so this correspondence is not fully symmetric.

4. Bridging

Given a mapping like the one described in Section 3, a common understanding of the two different technological spaces under consideration is given and a formalization of the commonalities and the variabilities of both worlds becomes feasible.

Bridging, i.e. combining different technological spaces, can be achieved in many different ways, depending on the context given for software development. But, bridging does not only pertain to the respective model and ontology *artifacts*, including their logical background and their languages. It has also to pertain to all *services and tools* belonging to the respective world, e.g., query and transformation languages.

Assuming, for instance, software development is done in some Modelware environment (e.g., in an Eclipse/Ecore/EMF world), it might make sense to use some specific Ontoware ontology for reusing some widely spread ontology for some special purpose (e.g., some domain ontology provided by the community). In this case, the use of that ontology by some API might be an appropriate approach. This approach uses only a loose coupling to the Ontoware world, from which only some well-defined services are used. We call this kind of bridging *adaptation*, since the API constitutes some kind of adapter that makes the ontology fit into the modeling world. But still both worlds coexist. Adaptation may also be used in the opposite direction.

If for some reason a pure homogeneous world is aspired, coexistence has to be replaced by some kind of *transformation* that generates a new model in the target space from a given model in the source space, e.g., a UML diagram may be used as a lightweight ontology in some Ontoware environment by generating a corresponding OWL ontology from it. Note, that this kind of bridging usually implies some loss of informa-

tion, since both worlds have different properties not all of which are transformable into the other world.

The third way of tackling the bridging of two different spaces is the *integration* of both spaces into one all-embracing new technological space. To achieve this, the mapping of concepts (which defines some kind of *intersection* of both worlds) has to be used to define a new target technological space which corresponds to the *union* of the source spaces. Apparently, such an integration affords a deep understanding (and thus a lot of research) and a good set of design decisions in the case of conflicting properties of both spaces.

In this section, we give short characterization of these three kinds of bridging approaches in terms of service definitions that specify the target result on an abstract level.

4.1. Adaptation

The first approach to bridging is keeping both technological spaces as they are, but make their *services* available in a common form. Given a mapping between the respective notions, a common service interface can be defined that acts as an adapter and allows an immediate comparison between both worlds.

The usefulness of a conceptual model depends on the services that come along with it. For application software that makes use of conceptual models, a *well-defined API* may grant use of such services in a generic way.

Figure 6 gives an overview on typical services supplied by conceptual models. The *specifications* given there are 'high-level' in the sense that they try to describe the services on a level of abstraction that (i) is precise enough to be understandable by experts and (ii) does not dive too much into the notational depth of a programming language. The specification assumes a class `ConceptualModel` which is an abstract superclass of classes `Model` and `Ontology` which implement the respective services.

Modelware offers models as versatile data structures that can be manipulated, analyzed, traversed, queried, and transformed into other models, whereas the focus in *Onto-ware* is on the use of reasoning services on the model that come along with description logics and make entailed knowledge available.

General Services. There are numerous tools that support *editing* of conceptual models in both worlds. Though there are more UML editors around than OWL editors, the integration of OCL-like constraints into UML editors is still not widely supported yet, whereas OWL includes the axiomatic parts because of the homogeneous character of description logics.

Query Answering is handled differently in both worlds. The type Query applies to some artifact that contains a query in some well-defined query language (e.g. OCL, GReQL, or SPARQL). *Models* can be queried by efficient query languages (depending on their internal representation) that are in the tradition of database technology. This holds for OCL [21] and especially GReQL [31] which is a graph query language. *Ontologies* are usually queried after a closed world completion by a reasoner which is an additional step that enlarges the ontology by making some entailed information manifest in the ontology. Thus, additional preprocessing is necessary. SPARQL [32] is a language for querying ontologies.

Consistency in general checks whether there is a non-empty instance graph for a model. This is a well-known reasoning service for ontologies, which is usually not of-

General Services.

Name	<i>Editing</i>
Signature	<code>void edit ();</code>
Description	supports external editing of the conceptual model <i>cm</i>
Name	<i>Query Answering</i>
Signature	<code>String query (Query q);</code>
Description	returns a string <i>s</i> which contains the result of applying the query <i>q</i> to the conceptual model <i>cm</i> (according to the semantics of the query language)
Name	<i>Consistency</i>
Signature	<code>boolean checkConsistency ();</code>
Description	checks whether the conceptual model <i>cm</i> is consistent
Explanation	a conceptual model <i>cm</i> is <i>consistent</i> iff there exists a non-empty instance graph for <i>cm</i> .
Name	<i>Instance Consistency</i>
Signature	<code>boolean checkInstanceConsistency (Instance i);</code>
Description	checks whether an instance graph <i>i</i> conforms to the conceptual model <i>cm</i>

Reasoning Services.

Name	<i>Satisfiability</i>
Signature	<code>boolean checkSatisfiability (Class c);</code>
Description	checks whether a class <i>c</i> is satisfiable in the conceptual model <i>cm</i>
Explanation	a class <i>c</i> is <i>satisfiable</i> (instantiable) in conceptual model <i>cm</i> iff there is an instance graph for <i>cm</i> with a non-empty set of vertices belonging to <i>c</i> . (Note, that satisfiability is a property of specific class. A conceptual model may contain non-satisfiable classes, while still being consistent.)
Name	<i>Subsumption</i>
Signature	<code>boolean subsumes (Class c1, Class c2);</code>
Description	checks whether a class <i>c1</i> subsumes a class <i>c2</i> in a conceptual model <i>cm</i>
Explanation	a class <i>c1</i> <i>subsumes</i> a class <i>c2</i> iff all instance vertices of <i>c2</i> are also instances of <i>c1</i> in all models.
Name	<i>LeastCommonSubsumer</i>
Signature	<code>Class getLeastCommonSubsumer (Set<Individual> s);</code>
Description	returns the smallest (named) class <i>c</i> that all individuals in <i>s</i> are instances of
Name	<i>Classification</i>
Signature	<code>boolean classifies (Class c, Individual i);</code>
Description	checks whether an individual <i>i</i> is an instance of a class <i>c</i> in a conceptual model <i>cm</i>

Figure 6. Services Offered as Methods by Conceptual Models

ferred for models. In contrast to that, *Instance consistency* (also called A-Box consistency) only checks an instance graph for its conformance to the model. This service is offered in both worlds, since it can be interpreted as a boolean query. In Ontoware it is usually done in an open world manner.

Reasoning Services. There are some services that rely on reasoning techniques and are usually not offered by models. Reasoning goes beyond querying in the sense that im-

Conceptual Model transformation.

Name	<i>Transformation Model to Ontology</i>
Signature	Ontology transform(Model m)
Description	transforms a model to a schema-like ontology
Name	<i>Transformation Ontology to Model</i>
Signature	Model transform(Ontology o)
Description	transforms the structural part of an ontology <i>o</i> to a model

Query transformation.

Name	<i>Transformation Model Query to Ontology Query</i>
Signature	OntologyQuery transform(ModelQuery q)
Description	transforms a model query <i>q</i> to an ontology query
Name	<i>Transformation Ontology Query to Model Query</i>
Signature	ModelQuery transform(OntologyQuery q)
Description	transforms an ontology query <i>q</i> to a model query

Figure 7. Transformation Service between Modelware and Ontoware

licit knowledge is inferred, whereas classical querying can refer to explicit knowledge only. *Satisfiability*, *subsumption*, and *classification* are typical ontology services that are usually not available on models.

Further Services. Besides the described services offered by conceptual models, there are more and more manipulation services offered today that act on them. With the advent of MDA, model *transformation* has become an important means for modeling. Also model *merging*, model *differencing*, and model *patching* are important for the support of model versioning. Though this work can be done for ontologies, as well, there seems to be not so much activity there in that respect, yet.

4.2. Transformation

Based on a mapping of elements of models and ontologies (Figure 5), it is quite easy to read UML-diagrams as schema-like ontologies. On the other hand, schema-like ontologies can also be translated to UML-diagrams as far as their structural part is concerned.

The transformation services specified in Figure 7 transform a *Model* according to a transformation definition which is based on a mapping like the one in Figure 5 to an *Ontology*. It creates for each class in a model a concept in the ontology. All associations and attributes are transformed to corresponding roles. Because UML class diagrams and ontologies contain instances of classes and concepts, respectively, objects of a model are transformed to individuals of an ontology. The transformation builds for each domain definition in a model a datatype in the ontology. For the subclassing relations in models, the transformation builds a subclass axiom in the ontology. The reverse transformation service realizes the translation from ontologies to models.

The translation of *axioms* in ontologies by the transformation service from ontologies to models in Figure 7 has to be treated more carefully. OCL-like axioms may use *set-theoretic constructs* based on an algorithmically feasible subset of predicate logic that may not be supported by the respective variant of the DL. On the other hand, class expressions in OWL (which define anonymous classes) may lead to *anonymous classes* in the UML world which might appear quite unnatural from the model point of view.

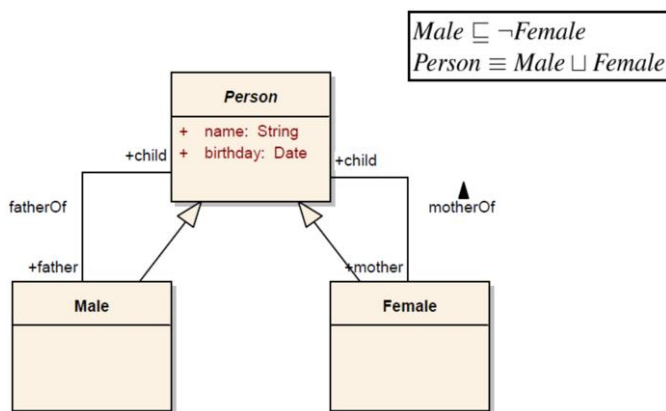


Figure 8. Hybrid Person Model

[33] describes a simple and a schema-aware translation from schema-like RDF graphs (which may be used for representing ontologies) to a graph-based variant of UML and backward.

Beside transformations of conceptual models a transformation bridge between model and ontology query languages can be established. The abstract specification of such transformation services is shown in Figure 7. [33] describes the mutual transformation of the respective query languages GReQL for models and SPARQL for ontologies.

Having a transformation service established, language users are able to build models with the modeling language (e.g. UML) and concrete syntaxes they are familiar with. The users are restricted to the expressiveness (e.g. different syntactic constructs) of the modeling language they are working with. The transformation services between query- and constraint languages increase the expressiveness of models to be transformed by the bridge, since they consider the models as well as annotated constraints or queries.

The tools and modeling environments for creating and processing models and ontologies remain the same. The transformation bridge establishes interoperability between modeling environments and different ontology tools (like reasoners).

4.3. Integration

The strongest form of bridging is the integration of both technological spaces into one single integrated one. Based on the mapping in Figure 5, it is possible in principle to integrate the *metamodels* of both worlds (assuming they are written in a common metamodeling language like MOF) by identifying (or subclassing) the respective notions. Then, the additional notions from both worlds may be added leading to a hybrid modeling language which could be able to use the best of both worlds.

A hybrid model conforms to a hybrid modeling language and is built by constructs of both languages. In the example in Figure 8 the UML class diagram from Figure 1 is extended by some DL axioms. It expresses, that the two concepts **Male** and **Female** are disjoint and the set of instances described by the concept **Person** is equivalent with the union of the two sets of instances described by **Male** and **Female**.

Figure 9 presents the basic services for language designers, building hybrid modeling languages, and for language users, using the hybrid modeling language.

To establish a hybrid modeling language the metamodels of both languages are combined. Figure 9 depicts an abstract specification of an integration services. It gets as input two metamodels and returns an integrated metamodel.

A concrete implementation of the integration service for the UML metamodel and the OWL metamodel affords a lot of design decisions, since comparability of notions does not immediately lead to a smooth integration. Language designers have to manually implement this metamodel integration service which results the integrated UML+OWL metamodel. Therefore they rely on a set of basic integration operations [34]. These operations allow for *merging* two classes, if the classes have the same meaning and are equivalent conceptually. They allow for *subclassing* two classes, if one class describes a more general concept than the other, or they just *establish an association* between two classes if the concepts are just related. Its use is based on intensional knowledge of the languages to be combined, and a mapping relating similar constructs of both worlds.

For the metamodels given in Figure 2 and 4, we suggest to merge UML class and DL concept, because in both technological spaces they have similar semantics, since both describe sets of instances. In addition, the UML association construct is combined with the DL concept role and the UML attribute is combined with the DL data role. Associations and concept roles describe sets of relations between instances. The attribute and the data role describe sets of relations between instances and data values.

An integrated metamodel is being developed in MOST. It is based on the TwoUse approach [35] and supplies a basis for conceptual modeling that combines the facilities of both worlds.

For language designers and users the interoperability with other tools is important. In particular, language users having created a hybrid model want to project it to a pure ontology which serves as input for several reasoning tools.

We propose the implementation of projection services. Figure 9 depicts the specifications of projection services. Both get as input a hybrid model (conforming to a metamodel designed by the integration service).

The ontology projection service projects the hybrid model to an ontology. Since we have integrated the UML class with the DL concept, the ontology projection service projects all classes in Figure 8 to pure DL concepts in the new ontology. In addition all associations and attributes depicted in Figure 8 are projected to DL concept roles and data roles. The additional concept constructors (for union and complement) conform to ontology constructs and are projected without any change to the new ontology.

The model projection service returns a new model which is built by the class diagram in Figure 8.

From the perspective of language users, the integration bridge provides a common view together on modelware and ontoware conceptual models, namely hybrid models. The modeling of hybrid models requires an understanding of both integrated languages. A language user has to be familiar with different concrete syntaxes (at least one for each modeling language) and how they are used in combination. To provide interoperability between different tools, projection services must be given by the integration bridge. Projection services extract all relevant information from hybrid models and translate them to models understandable by given tools. For example hybrid models are projected to ontologies to be readable by reasoners.

Abstract Integration Service.

Name	<i>Integration of Metamodels</i>
Signature	Metamodel integrate(Metamodel m_u , Metamodel m_o)
Description	integrates two metamodels m_u and m_o . The result is an integrated meta-model.

Integration Operations.

Name	<i>Merge Integration Service</i>
Signature	Class merge(Class c_1 , Class c_2)
Description	merges the two classes c_1 and c_2 by replacing them by a new class. All incidences with associations and specialization relations and all nested attributes are moved to the new class.

Name	<i>Specialize Integration Service</i>
Signature	void specialize(Class c_1 , Class c_2)
Description	creates a specialization relationship between two classes c_1 and c_2 .

Name	<i>Associate Integration Service</i>
Signature	Association associate(Class c_1 , Class c_2)
Description	associates two classes c_1 and c_2 by a newly created association

Projection Services.

Name	<i>Ontology Projection Service</i>
Signature	Ontology project _o (ConceptualModel m_c)
Description	creates a new ontology m_o which only consists of those parts of m_c which conform to ontology constructs defined in the metamodel of an ontology language (e.g the one in [9]).

Name	<i>Model Projection Service</i>
Signature	Model project _m (ConceptualModel m_c)
Description	creates a new model m_m which only consists of those parts of m_c which conform to UML class diagram constructs defined in the class diagram meta-model.

Figure 9. Integration and Projection Services**5. Discussion**

A comparison between Modelware and Ontoware is always subjective to some extent since the results have been developed in different scientific communities with different goals and different terminologies. In essence, the support used by a bridging approach has to be chosen carefully depending on the features needed.

5.1. Comparing Modelware and Ontoware

As the descriptions above have shown comparison of Modelware and Ontoware is a multi-faceted issue which has several dimensions: (a) logics, (b) languages, (c) tools, and (d) conceptual modeling styles. Thus, depending on the situation and on the requirements of the actual application different combinations of logics, languages, tools, and styles might be appropriate.

(a) logics: Modelware and Ontoware both provide declarative means to describe conceptual models. While Modelware is based on *constructive set theory*, whose descrip-

tions consist of a structural part and additional constraints, Ontoware is based on *description logic*, which allows the joint description of both parts. The subtle difference entailed by the open world assumption in ontologies (as opposed to the closed world assumption in models) leads to different results e.g. for consistency checking, which may be quite useful but has to be used with care.

(b) languages: To come to concrete statements about similarities and dissimilarities between both worlds, concrete languages have to be chosen for describing structure and constraints for models on the one hand and for the ontologies on the other hand. As DL theory suggests, several variants of descriptions logics may be used with differing properties. The OWL 2 family of ontology languages offers several such profiles.

(c) tools: Both worlds are supported by tools. There are *visual editors* for models (various UML tools) and *textual editors* for ontologies (e.g. protégé). Since models also act as prescriptions, *code generation tools* are available, as well. There seem to be much more manipulation tools around on the Modelware side especially stimulated by the needs of MDA. This also pertains to the respective query and manipulation languages (for transformation, differencing, patching, etc.). The prescriptive aspect of ontologies primarily pertains to their *reasoning facilities*, when they are used via their runtime services. While classical modeling is more efficient for querying and A-Box constraint checking, ontologies supply simultaneous T- and A-Box checking and open world reasoning.

(d) styles: The concrete editing, querying and reasoning *technologies* applied surely imply different modeling styles and the use of different services. In general, Modelware languages have a stronger typing schema and come with more constraints on the model structure than general predicates. Therefore more disciplines and conventions are necessary to keep ontologies well-structured. Especially for ontologies, the *modeling style* used influences the comparison. As stated above, schema-like modeling eases the use of class diagrams as a lightweight description of the concept structure and allows a schema-aware transformation of query languages.

On the other hand, there is a well-developed body of knowledge for modeling styles, including collections of *design patterns*, catalogs of *best practice*, and work on the evaluation of *quality of models*, which is not directly usable for the corresponding ontologies.

5.2. Comparing Bridging Approaches

To bring both worlds together, this paper distinguishes three different bridging approaches, namely (a) adaptation, (b) transformation, and (c) integration. Each of these approaches has different advantages and disadvantages.

(a) adaptation: The definition of application programming interfaces (APIs) which make the common usage of Modelware and Ontoware facilities possible is the current state of the practice. All reasoners supply such an API for the construction of ontologies and the respective reasoning services. But still the interoperability between tools for constructing and editing models or ontologies, respectively, and application software that makes use of these models at runtime, is far from being standardized.

(b) transformation: Tools for transforming conceptual models and their corresponding constraints and queries from one world into the other are quite helpful for many applications. But the fact that both worlds have properties that are not directly present in the other world shows the limitations of this approach.

(c) integration: The probably best way to bridge both worlds seems to be an amalgamation of Modelware and Ontoware. The common kernel of both worlds sketched in this paper shows that it should be possible to define support for conceptual modeling which has the strength of schema-like modeling including full constraint checking (from Modelware) as well as the facilities of reasoning (from Ontoware). Though the fact that both worlds are well-established with their own communities, languages, techniques, and tools makes such a development hard, following the presentation from above such an integration still seems to be accomplishable.

6. Conclusion

This paper gave some impressions on the comparison of two different technological spaces for conceptual modeling, namely the UML/MOF world, called *Modelware* for short, and the OWL world, called *Ontoware*. It turned out that the comparison of Modelware and Ontoware is a multi-faceted issue, since there are many alternative features to choose from on both sides.

Though bridging seems reasonable in principle, there are still subtle differences in the use of both technological spaces which also were shortly discussed. It turned out that a definite decision is only possible if a multidimensional comparison is done on concrete implementations.

The work presented in this paper shows that in principle it is possible to create a common technological space supplying the united power of Modelware and Ontoware.

Simultaneously, looking at the diversity of languages, techniques and tools and the quite small amount of research done in uniting both worlds as well as the quite disjoint communities and the different consortia (OMG and W3C) there is still a long way to go.

Acknowledgement

The author would like to thank the MOST team for the productive working atmosphere in this project. Most of the statements in this paper have their roots in discussions with the consortium. Special thanks go to the Koblenz team, especially to Hannes Schwarz, Steffen Staab, Fernando Parreiras and Gerd Gröner.

References

- [1] Kurtev, I., Bézivin, J., Aksit, M.: Technological spaces: An initial appraisal. In: International Symposium on Distributed Objects and Applications, DOA 2002. (2002)
- [2] Ebert, J.: Software engineering with models and ontologies. In Barzdins, J., Kirikova, M., eds.: Databases and Information Systems, Riga, University of Latvia Press (2010)
- [3] Walter, T., Parreiras, F.S., Staab, S., Ebert, J.: Joint Language and Domain Engineering. In: Proc. of 6th European Conference on Modelling Foundations and Applications, ECMFA 2010, Paris. Volume 6138 of LNCS., Springer (2010)
- [4] Walter, T., Silva Parreiras, F., Staab, S.: OntoDSL: An Ontology-Based Framework for Domain-Specific Languages. In: ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems, 12th International Conference, MODELS 2009. Volume 5795., Springer (2009) 408–422
- [5] Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition **6** (1993) 199–221

- [6] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Pater-Schneider, P., eds.: *The Description Logic Handbook*. Cambridge University Press, Cambridge (2003)
- [7] OMG: UML 2.0 Superstructure Specification (formal/05-07-04). Technical report (2005)
- [8] Baader, F., Nutt, W.: Basic description logics. In: *The Description Logic Handbook*. Cambridge University Press (2003)
- [9] W3C, ed.: *OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax*. (2009)
- [10] Chen, P.P.S.: The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.* **1**(1) (1976) 9–36
- [11] Verheijen, G.M.A., van Bekkum, J.: Niam: An information analysis method. In Olle, T.W., Sol, H.G., Verrijn-Stuart, A.A., eds.: *Information System Methodologies*, Amsterdam, North-Holland (1982)
- [12] Martin, J., McClure, C.: *Diagramming Techniques for Analysts and Programmers*. Prentice-Hall, Englewood Cliffs (1985)
- [13] Booch, G.: *Object-Oriented Design*. Benjamin/Cummings, Redwood City (1991)
- [14] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs NJ (1991) Original.
- [15] Jacobson, I.: *Object-Oriented Software Engineering*. Addison-Wesley, Wokingham (1992)
- [16] Junginger, S., Kühn, H., Strobl, R., Karagiannis, D.: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen. *Wirtschaftsinformatik* **42** (2000) 392–401
- [17] Muller, P.A., Fleurey, F., Jézéquel, J.M.: Weaving executability into object-oriented meta-languages. In: *MODELS 2005*. (2005)
- [18] Ebert, J., Riediger, V., Winter, A.: Graph Technology in Reverse Engineering, the TGraph Approach. In Gimnich, R., Kaiser, U., Winter, A., eds.: *10th Workshop Software Reengineering (WSR 2008)*. Volume 126 of *GI Lecture Notes in Informatics*., Bonn, GI (2008) 67–81
- [19] OMG: MDA guide version 1.0.1. Technical report (2003)
- [20] OMG: MOF Query / Views / Transformations. Technical report (2008)
- [21] OMG: Object Constraint Language, v2.0. Technical report (2006)
- [22] Walter, T., Ebert, J.: Foundations of graph-based modeling languages. Technical report, Dept. of Computer Science, University Koblenz-Landau (to appear, 2010)
- [23] Taentzer, G.: AGG: A graph transformation environment for modeling and validation of software. In Pfaltz, L., Nagl, M., Böhlen, B., eds.: *Applications of Graph Transformations with Industrial Relevance: Second International Workshop (AGTIVE 2003)*, Springer (2004) 446–453
- [24] Ranger, U., Weinell, E.: The graph rewriting language and environment progres. In: *Applications of Graph Transformations with Industrial Relevance*. vol. 5088 of *LNCS*, Springer (2008)
- [25] Kalnins, A., Barzdins, J., E.Celms: Model transformation language MOLA. In: *Proceedings of Model-Driven Architecture: Foundations and Applications (MDAFA 2004)*. (2004) 14–28
- [26] Horn, T., Ebert, J.: The GReTL transformation language. Technical report, Department of Computer Science, University Koblenz-Landau, Koblenz (2010)
- [27] Schmitt, P.H.: A model theoretic semantics of OCL. (2001)
- [28] Kifer, M., Lausen, G.: F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. *SIGMOD Rec.* **18**(2) (1989) 134–146
- [29] W3C, ed.: *Resource Description Framework (RDF): Concepts and Abstract Syntax*. (2004)
- [30] W3C, ed.: *OWL 2 Web Ontology Language Profiles*. (2009)
- [31] Ebert, J., Bildhauer, D.: Reverse engineering using graph queries. In Schürr, A., Lewerentz, C., Engels, G., Schäfer, W., Westfechtel, B., eds.: *Graph Transformations and Model Driven Engineering*. LNCS 5765. Springer (2010) to appear.
- [32] W3C, ed.: *SPARQL Query Language for RDF, W3C Recommendation*. (2008) <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>.
- [33] Schwarz, H., Ebert, J.: Bridging Query Languages in Semantic and Graph Technologies. In: *Reasoning Web. Semantic Technologies for Software Engineering*. Volume 6325 of *LNCS*., Springer (2010)
- [34] Walter, T., Ebert, J.: Combining DSLs and Ontologies using Metamodel Integration. In: *Domain-Specific Languages*. Volume 5658 of *LNCS*., Springer (2009) 148–169
- [35] Silva Parreiras, F., Staab, S.: Using Ontologies with UML Class-based Modeling: The TwoUse Approach. *Data and Knowledge Engineering* ([accepted for Publication], 2010)

Requirements-Level Programming for Rapid Software Evolution

Michał ŚMIAŁEK ^a

^a *Warsaw University of Technology, Warsaw, Poland*

Abstract. Rapid development of evolving software systems is highly associated with the ability to react quickly to changing user requirements. This paper presents a coherent set of technologies for simplifying the path from evolving requirements to code. The most important novel element on this path is a language defined at the level of requirements (understandable for non-IT experts) that is equipped with operational semantics. This means that it is possible to translate specifications written in this language, automatically into executable code. The language also allows for easy detection of changes in requirements. This detection can be propagated down to the code structure and appropriate code parts (these that are not automatically generated) indicated for rework. It will be demonstrated that the presented approach is effective and suitable for a wide range of problem domains as opposed to domain-specific approaches. This will be shown through a case study for a typical business software system, performed with a novel tool suite.

Keywords. requirements, use cases, model transformations, code generation

Introduction and related work

Evolution of software is always associated with performing changes to code. Despite different natures of these changes (extending functionality, refactoring, correcting, etc.), the source is always the same: requirements. Changes to requirements cause changes to the system.

In a rapidly changing world, it is important that software can react quickly to changes. This can be done by reducing the effort to “translate” requirements into the final system. Traditionally, this is done through a manual process of analysing natural language text and based on this analysis - designing the system and producing code. In the recent years, there were introduced prominent approaches to automate this path. In Model-Driven Architecture (MDA; see [1]) there was introduced the concept of several modelling layers at different levels of abstraction. The first level is independent of the computations (cf. requirements), and the subsequent layers add certain design details leading to the final code. Models at each of the layers are produced in a series of automatic transformations. This leads to organising software development around artifacts that are models (see a wide overview in [2]).

Model Driven Software Development introduces at least partial automation in reaching code from higher level models. In Domain-Specific Modelling, this is shifted even further, so that the production of code is fully automated (see eg. [3] for a good overview).

This way, models become the actual code. However, this is achieved at a price of reducing the applicability of a given modelling language and associated models, to just one, relatively limited problem domain (cf. domain specificity). This means that models produced for one class of systems cannot be reused for other kinds, and the software developers cannot shift easily from one problem domain to another. Domain-Specific Modelling is closely related to automated Software Product Lines (see [4]) or Software Factories (see [5]). In this approach, software is built in an automated way so that a series of similar (evolving) systems can be developed. Every new system is built as a variant of a base system and only some changes to the intermediate models have to be made. Software product lines and software factories are part of a more general trend to organise software evolution (see [6] for a wide overview of this field). In general, we would like to automate the process in which software changes. Unfortunately, most approaches concentrate on the evolution of design artifacts and code. Requirements are treated as second-class citizens in this process. Requirements-based evolution of software is not a frequent subject of research (see [6] for its position within the overall research area and [7] for the current state-of-the-art in requirements evolution).

In [7] it is proposed that a comprehensive tool infrastructure for requirements evolution is built. This paper replies to this research direction in proposing an infrastructure for treating requirements as first-class artifacts. This means that requirements are introduced directly into the path of automatic transformations. This is achieved by structuring natural language and giving it runtime semantics. The proposed infrastructure includes automatic transition from domain-independent requirements to design and code. It can be noted that also some previous approaches treat the issues of interplay between requirements and design (see [8] and [9] for important insights). In [10], a process for evolving requirements with separate domain models and associated implementation is presented. Unlike for our approach, automatic transformations and pointers to necessary changes in code are not handled. In [11], theoretical foundations for the evolution of requirements (resulting in their well-formedness) are introduced. Traceability of evolving requirements is also discussed, although again, no automation is introduced. Requirements evolution can also be related to product family construction which is based on requirements variability analysis (see eg. [12]).

This paper follows and significantly extends the process presented in [13] which advocates high levels of automation on the path from requirements to code. We present a requirements specification language where the most detailed specifications can be treated as the first step towards specifying the problem solution (as opposed to the problem definition), which is discussed in [14]. This first step is then translated into code that evolves together with requirements (see [15] for a relevant insight). In the following sections we present the syntax of an appropriate requirements specification language and introduce its runtime semantics. This is an important contribution of this paper where a general-purpose language at a high level of abstraction (much higher than the current general purpose programming languages) has its runtime semantics defined. This definition is conducted by giving precise rules for transforming specifications in this language into compilable code. Based on this runtime capabilities we also present a schema that introduces high levels of automation into software evolution controlled through requirements.

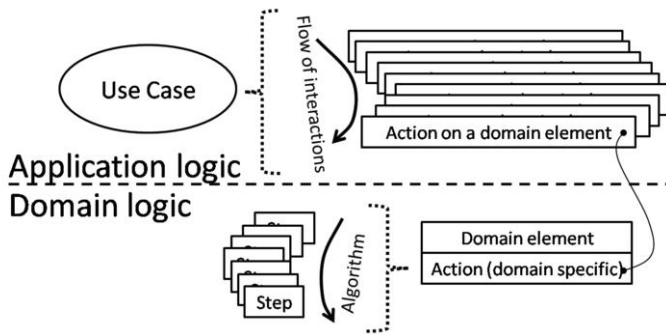


Figure 1. Application logic and domain logic layers of the presented language

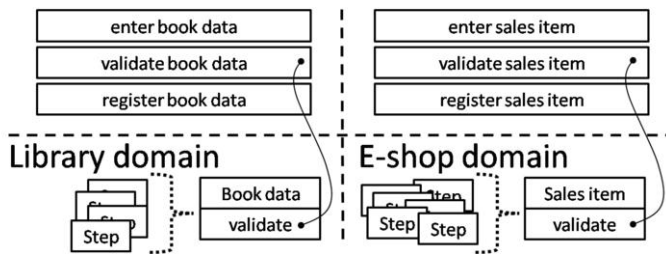


Figure 2. Hyper-linking the application with the domain

1. Programming at the requirements level

In Domain-Specific Modelling, models become the de-facto code. This approach of shifting the level at which programming is done allows for achieving very significant raise in productivity (as pointed out in [3]). This is, though, achieved at a cost of learning and maintaining self-designed modelling languages. We thus would like to design a general-purpose modelling language suitable for various problem domains. At the same time, this language should be understandable by domain experts (especially including non-IT specialists) who participate in specifying requirements for software systems.

For the above reasons, we will base our design of a requirements-level programming language on a widely accepted approach to specifying functional requirements. We will apply use cases (see [16] for the initial idea) which are also part of the widely used Unified Modelling Language (see [17] for the official specification). Use cases are normally not seen as first-class modelling artifacts and do not normally participate in the model transformation paths mentioned in the previous section. This is because the use case representations have quite vague semantics and approaches to clarify this are sparse (see [18] for one of the few). In [19] there was proposed an extension to the use case model which clarifies the control flow of use cases and facilitates its automatic transformation into other modelling artifacts.

The general idea of requirements-level programming is presented in Figure 1. The use case representations define flows of user-system interactions which specify the application logic of the system. This application logic uses terms and phrases specific for a given problem domain. These terms (domain elements) and phrases (domain-specific actions) constitute the domain logic with appropriately defined processing algorithms.

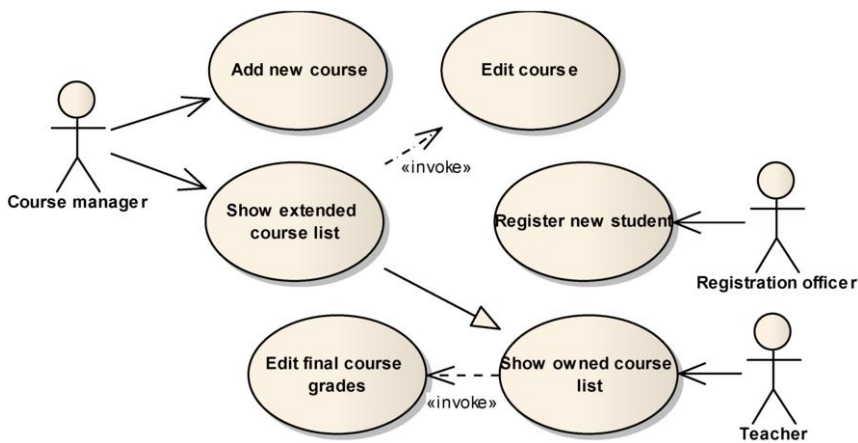


Figure 3. Initial use case model of the case study

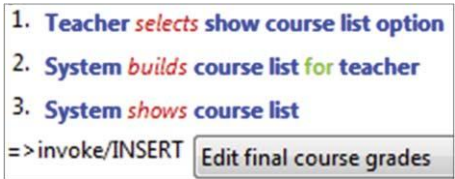


Figure 4. Scenario of the “Show owned course list” use case

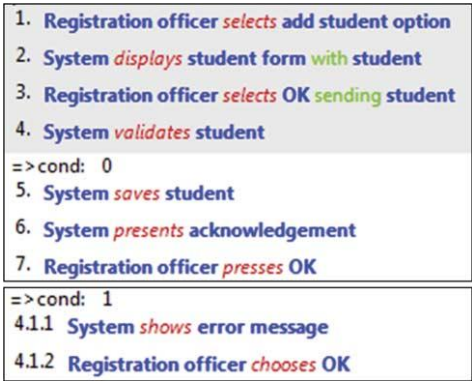


Figure 5. Scenarios of the “Add new course” use case

What is important, the domain logic (cf. domain-specific languages) is declared in a language which does not depend on any specific domain. This way we result in a general-purpose language where the application logic is precisely separated from the domain logic. It can be noted that the specifications at the application logic level can be easily applied to any other problem domain by substituting terms and phrases. This is illustrated in an example in Fig. 2. There we can see two identical fragments of application logic applied to two different problem domains with different processing algorithms.

The above presented criteria for a programming language at requirements level are

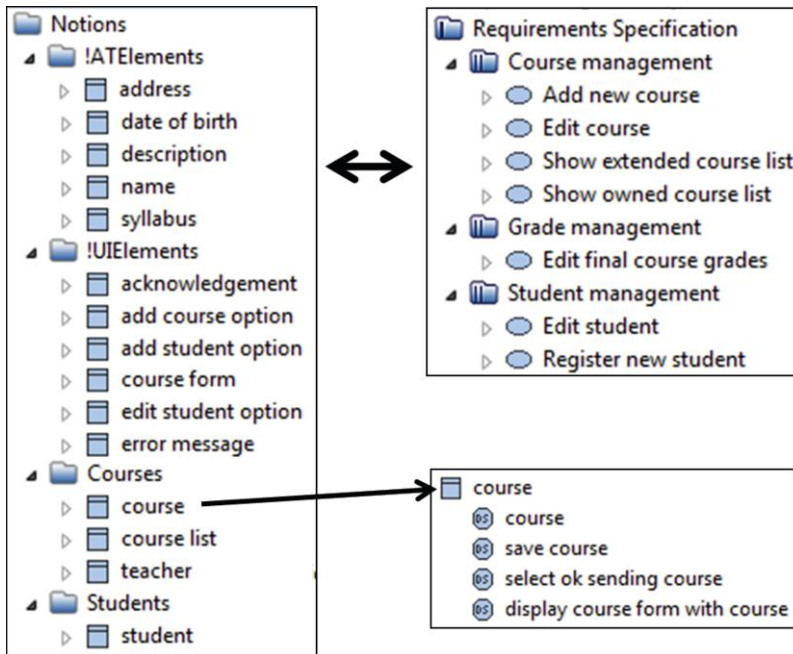


Figure 6. Structure of the initial requirements model

met by the Requirements Specification Language (RSL, see [20] for full specification) developed within the ReDSeeDS project (www.redseeds.eu). The language allows for specifying use case representations, defining flows of events and associating terms used in these events with a central domain vocabulary. We will illustrate the language through a case study example. In Figure 3 we can see a fragment of the use case model of a Campus Management System (CMS). This model follows standard use case modelling principles with the exception of two $\ll\text{invoke}\gg$ relationships. This type of relationship substitutes the ambiguous $\ll\text{extend}\gg$ and $\ll\text{include}\gg$ relationships (see [21]). Figures 4 and 5 show the details of use case representations. The presented scenarios are written in a simple subject-verb-object (SVO) grammar. They also show how the $\ll\text{invoke}\gg$ relationship is weaved into the flow of events. In Figure 5 we can see two scenarios with their flow being controlled by two conditions ($=\gg$ cond). It can be also noted that the objects in the SVO are either elements of the problem domain (e.g. “student”) or the user interface (e.g. “student form”). This distinction will help in generating code as described in the further sections.

The SVO sentences are linked to a central vocabulary which is presented in Figure 6 (left). Every object in a sentence has an associated domain element (notion) in the vocabulary. Obviously, objects in different sentences can point to the same domain element. This way, the whole specification is coherent with scenarios written in a uniform terminology. At the same time, the vocabulary offers space for specifying the domain logic (placed under domain phrases like “save course”). It can be noted, that an additional activity from the requirements specifiers is to divide both the use case model and the domain vocabulary into packages. These packages will be used to structure the final code generated from requirements, as presented in the next section.

2. From requirements to code

In order to be able to generate code from the above presented requirements models we need to assign operational semantics to them. This would mean that each of the RSL constructs would have their meaning explained in the code execution environment. We will supply this meaning by specifying automatic rules for transforming these constructs into Java-like code. This way, the RSL operational semantics will be transformed onto Java runtime semantics. The following list presents the most important rules (some rules were omitted for brevity), concentrating on the elements defining the application logic.

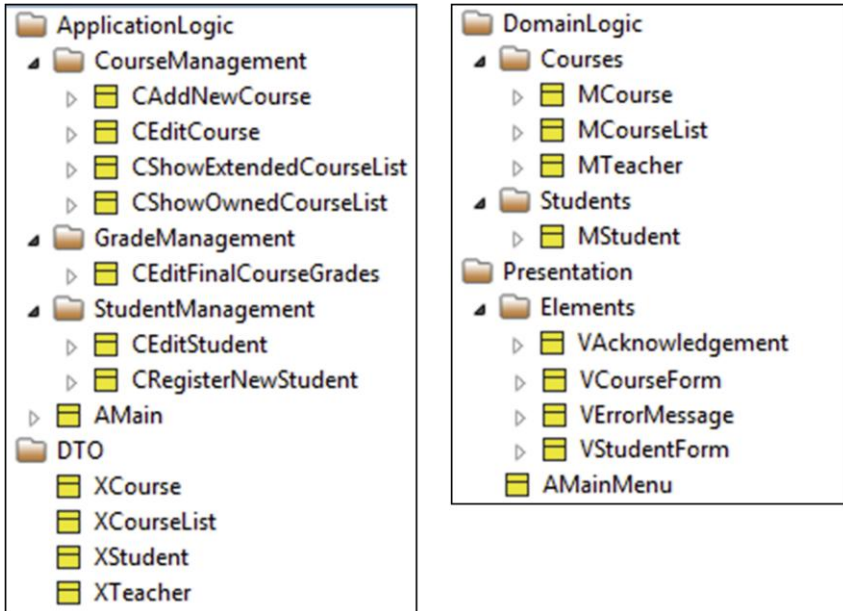


Figure 7. Generated general structure of the case study system (rules 1-3)

1. **Package transformation.** Every package in the requirements model is transformed into a package in Java. The packages are divided into the application logic layer (use case packages) and the domain logic layer (notion packages). In addition, a single package for user-interface-related operations is generated.
2. **Use case transformation.** Every use case is transformed into a public class (with appropriately concatenated name) in the associated application logic layer package. The class name is derived from the use case name (“camel case”) with the “C” (Controller) prefix.
3. **Notion transformation.** Every notion is transformed into a public class (with appropriately concatenated name) in the associated domain logic package. The class name is derived from the use case name (“camel case”) with the “M” (Model) prefix.
4. **Phrase transformation.** Every verb phrase in the domain model is transformed into an operation in the class generated either from the containing notion or from the use cases that use this phrase. The name of the operation is a concatenation of words of the phrase. The detailed rules are as follows.

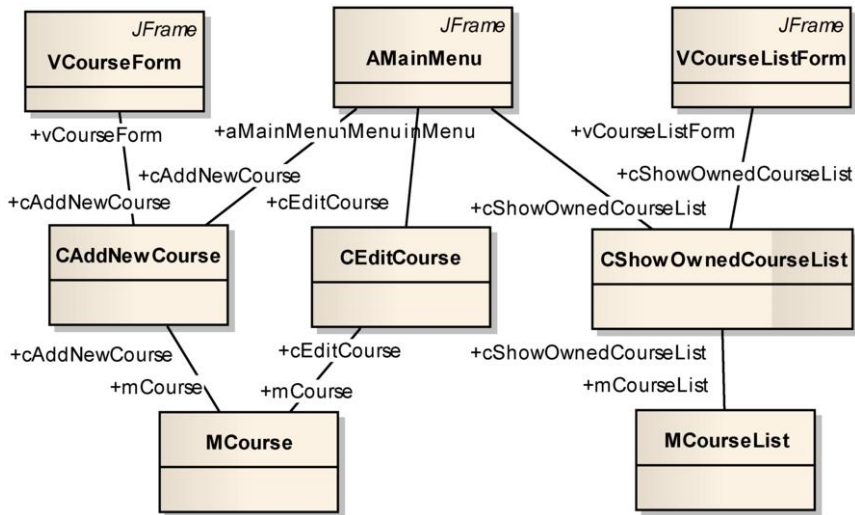


Figure 8. Illustration of rules 1-3 applied to the CourseManagement package

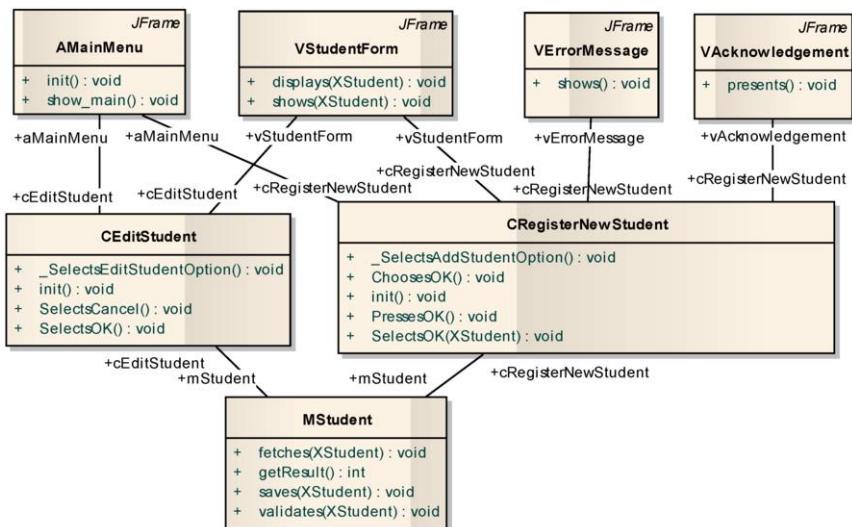


Figure 9. Generated classes and their operations (rules 2-4) for the StudentManagement package

- Verb phrase used in an SVO sentence where an actor is the subject and a domain element is the direct object is generated into an operation of the class generated from the use case containing the SVO sentence.
- Verb phrase used in an SVO sentence where the system is the subject and a user interface element is the direct object is generated into an operation of the class within the user-interface-related package.
- Verb phrase used in an SVO sentence where the system is the subject and a domain element is the direct object is generated into an operation of the class generated from the containing notion.

```

package App.ApplicationLogic.CourseManagement;
public class CShowOwnedCourseList {

    public XCourseList aCourseList;
    public XTeacher aTeacher;
    public MCourseList mCourseList;
    public VCourseListForm vCourseListForm;

    public void _SelectsShowCourseListOption() {
        int res;
        mCourseList.builds(aCourseList,aTeacher);
        res = mCourseList.getResult();
        vCourseListForm = new VCourseListForm();
        vCourseListForm.cShowOwnedCourseList = this;
        vCourseListForm.shows(aCourseList);
    }
}

```

Figure 10. Generated dynamic code for “Show owned course list” (rules 5a-5c)

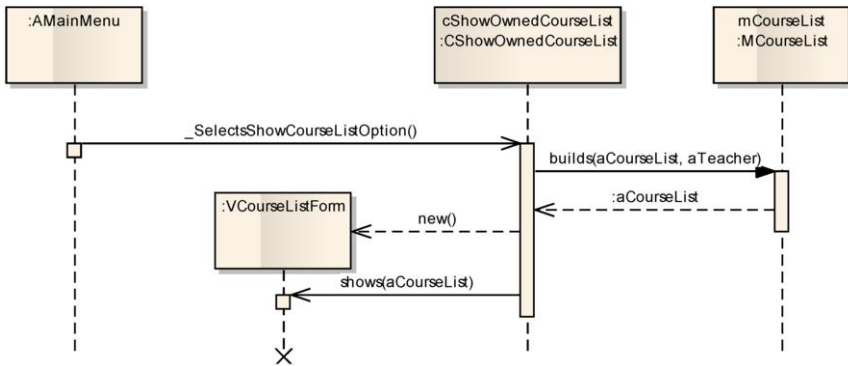


Figure 11. Sequence diagram explaining code from Fig. 10

5. **Scenario transformation.** Every set of scenarios for a use case is transformed into code within the methods of the class generated from this scenario. The detailed rules are as follows.

- Code of the class generated from a use case contains methods associated with operations generated according to rule 4a.
- Every method in the above class contains code generated from sentences that follow an SVO sentence from rule 4a and finish at the next such sentence.
- Inside the above method, there are generated method calls. For an SVO sentence from rule 4b, a call to the method realising a user-interface-related operation is generated. For an SVO sentence from rule 4c, a call to the method realising a domain logic operation is generated.
- Alternatives in scenarios are transformed into conditional instructions (“if”). For each of the alternative scenarios, a branch in the conditional instruction is generated. The branching is done by detecting the value returned by the method call generated from the last SVO sentence before the alternative in the scenario.

The above rules are illustrated in the case study example. Figures 7 to 13 show the structure and dynamics of the code generated from the requirements model presented in

```

package App.ApplicationLogic.StudentManagement;
public class CRegisterNewStudent {
    public XStudent aStudent;
    public VStudentForm vStudentForm;
    public MStudent mStudent;
    public VAcknowledgement vAcknowledgement;
    public VErrorMessage vErrorMessage;

    public void _SelectsAddStudentOption(){
        int res;
        vStudentForm = new VStudentForm();
        vStudentForm.cRegisterNewStudent = this;
        vStudentForm.displays(aStudent);
    }

    public void SelectsOK(XStudent pStudent){
        int res;
        aStudent = pStudent;
        mStudent.validates(aStudent); res = mStudent.getResult();
        if (res == 0) {
            mStudent.saves(aStudent); res = mStudent.getResult();
            vAcknowledgement = new VAcknowledgement();
            vAcknowledgement.cRegisterNewStudent = this;
            vAcknowledgement.presents();
        } else if (res == 1) {
            vErrorMessage = new VErrorMessage();
            vErrorMessage.cRegisterNewStudent = this;
            vErrorMessage.shows();
        }
    }
}

```

Figure 12. Generated dynamic code for “Add new course” (rules 5a-5d)

the previous section. Figure 7 presents the application of rules 1 to 3. It shows the packages and classes generated from the requirements model structure shown in Figure 6. In Figure 8 we can see some more details for the part of the system that participates in the functionality of “Course management”. This UML class diagram shows three layers with the “C” classes defining the application logic (controllers) for the appropriate use cases. The “M” classes are associated with the appropriate “C” classes and form the domain logic (models) of the system. Also the “V” (view, or user interface) classes are generated according to the scenarios and associated with appropriate controllers. In summary, each of the packages of the application logic layer has classes generated from use cases (e.g. the “Edit course” use case is transformed into the CEditCourse class). Similar rule was applied to the domain logic layer (e.g. the “course list” notion transformed into the MCourseList class).

More detailed example for rules 2, 3 and 4 is presented in Figure 9. There we can see the classes associated with the “Student management” functionality. In this class diagram we have revealed also the details of the class operations. It can be noted that the operations of classes CEditStudent and CRegisterNewStudent are generated according to rule 4a. Moreover, we can see the MStudent class where its operations are compliant with the rule 4c. Finally, there are presented three “V” classes that define the user interface frames. Their operations were generated according to rule 4b. The above classes and their operations are used to fulfill the scenario presented in Figure 5.

Figures 10 and 12 show the most important parts of code of two “C” classes. This code has been generated using all five rules. Thus, it includes also the dynamic part (the method contents). The associated UML interaction (sequence) diagrams explain how

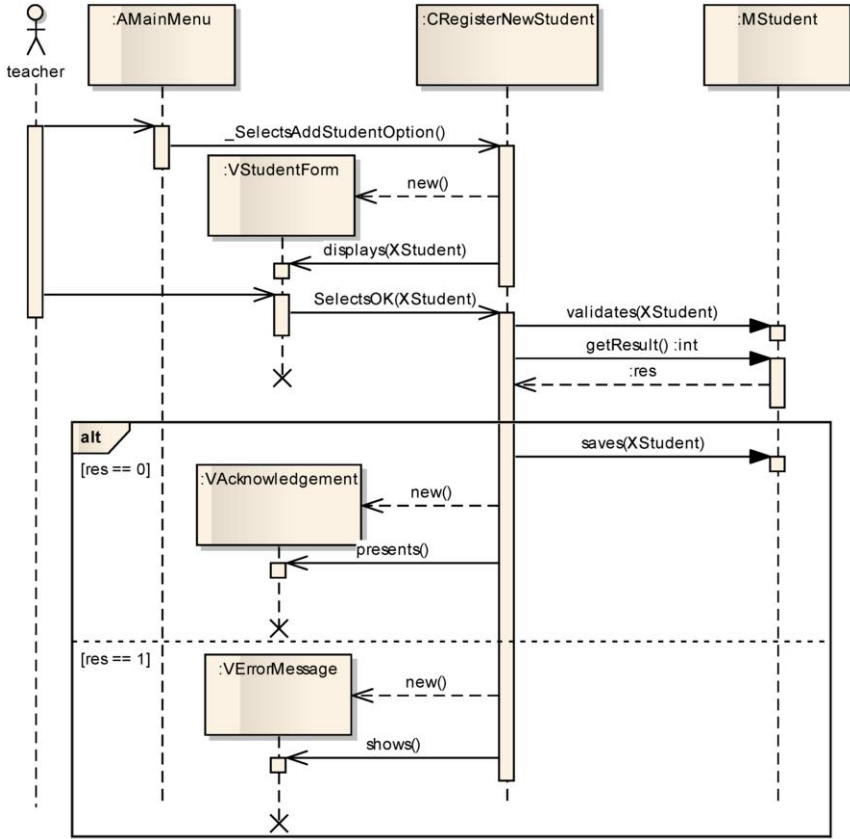


Figure 13. Sequence diagram explaining code from Fig. 12

this code is equivalent to scenarios presented in Figures 4 and 5. Comparison of these four figures should facilitate understanding of the set of rules 5a-5d. It can be noted that in order to generate all the code details (like e.g. the call/method signatures), some additional markings in the requirements model and additional rules would need to be applied. These simple rules were omitted due to limited space.

3. Software evolution as requirements evolution

Having the automatic code generation mechanisms presented in the previous section we can now define a schema for software evolution. As postulated in the introduction, the schema is driven by changes in requirements. This is illustrated in Figure 14. It shows two iterations in an evolutionary software lifecycle. In the first iteration, the requirements (describing the application logic R_{AL1} and the domain logic R_{DL1}) are specified and appropriate code generated. According to what was demonstrated in the previous section, the application logic code C_{AL1} can be generated fully, but the domain logic code C_{DL1} can be generated partially (only the class structure, methods and signatures). In the second iteration, both the application logic and domain logic requirements change (δ_{RAL} and δ_{RDL}). After the change, the application logic code C_{AL2} can be generated

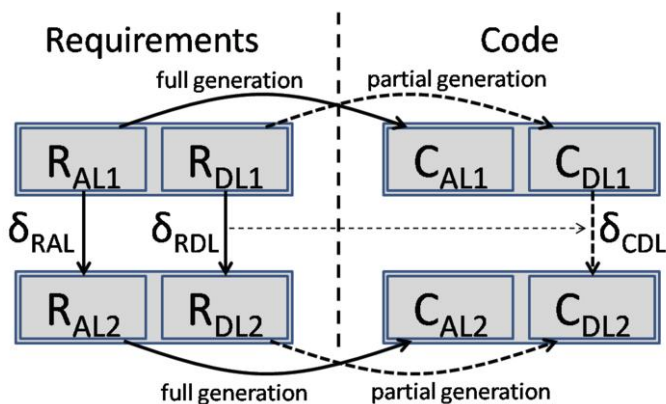


Figure 14. General schema for software evolution

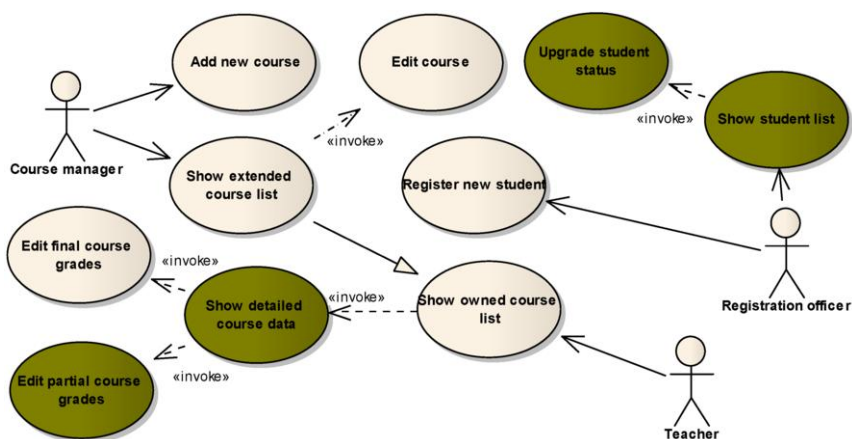


Figure 15. Updated use case model of the case study

automatically, and no additional effort is needed. However, there remains a crucial issue of determining changes to the domain logic code C_{DL2} . We would like to know which methods in the partially generated code C_{DL2} need updates by programmers and to what extent. This can be done by examining changes in domain logic requirements δ_{RDL} . It can be noted that in order to examine these changes (especially in large systems with vast requirements specifications) we need a tool which compares two requirements specifications and shows the δ . By tracing this delta into code, the methods needing rework can be indicated very precisely (δ_{CDL}).

Let us now examine the schema we have sketched above, within our case study example. The changed requirements are presented in Figures 15 and 16. The first diagram shows the updated use case model, with added use cases highlighted. The second diagram presents changes made to the representation of one of the use cases. These changes also indicate additional user interface functionality necessary for fulfilling this additional application logic. Namely, in this new version, the system needs to “display again” the student form which assumes also displaying an additional error message and a request to correct the student data.

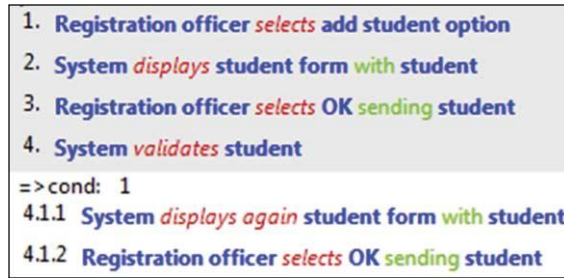


Figure 16. Updated scenario of the “Show owned course list” use case

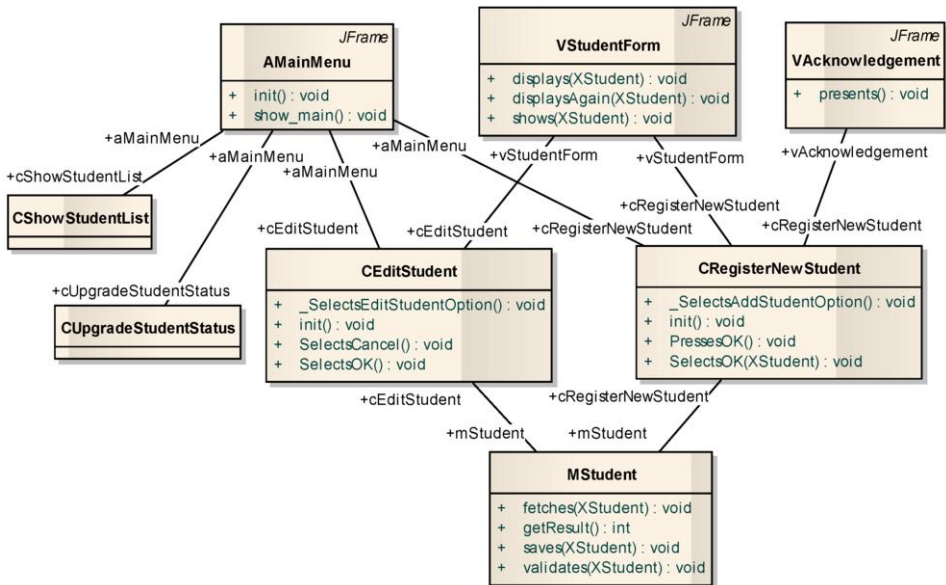


Figure 17. Updated structure of the case study system

Based on these changed requirements, code has been re-built. Its new version now has an updated structure shown in Figure 17. It can be noted that the structure has been extended appropriately to the extension in the use case set and the necessary domain elements (notions). New classes (CShowStudentList, CUpgradeStudentStatus) have been added, and the existing classes have been modified (only the relevant part of the system was shown for brevity). By comparing Figures 9 and 17 it can be easily determined which additional operations have been created for the new logic.

Figure 18 show updates made to the code of the CRegisterNewStudent class. Changes can be easily determined by comparing this with Figure 12. It can be noted that the code has been updated by calling the additional “displaysAgain” method. In Figure 19 we can additionally see an excerpt of the code generated within the presentation layer. This code forms an event handler that reacts to pressing the “OK” button on the “student form” window, as shown in Figure 20. The code for showing the window and the event handler could also be generated automatically. This was done according to additional

```

public class CRegisterNewStudent {
    public void SelectsOK(XStudent pStudent){
        int res;
        aStudent = pStudent;
        mStudent.validates(aStudent);
        res = mStudent.getResult();
        if (res == 0) {
            mStudent.saves(aStudent); res = mStudent.getResult();
            vAcknowledgement = new VAcknowledgement();
            vAcknowledgement.cRegisterNewStudent = this;
            vAcknowledgement.presents();
        } else if (res == 1) {
            vStudentForm = new VStudentForm();
            vStudentForm.cRegisterNewStudent = this;
            vStudentForm.displaysAgain(aStudent);
        }
    }
}

```

Figure 18. Updated application logic code

```

public class VStudentForm extends JFrame {
    public void displays(XStudent aStudent){
        // ...
        JButton jbnOK = new JButton("\u0020OK\u0022");
        jbnOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                XStudent aStudent = new XStudent();
                aStudent.address = jtfaddress.getText();
                aStudent.name = jtfname.getText();
                aStudent.dateOfBirth = jtfdateOfBirth.getText();
                cRegisterNewStudent.SelectsOK(aStudent);
            }
        });
        // ...
    }
}

```

Figure 19. Sample code of the presentation layer class



Figure 20. Example window generated automatically from the requirements

rules, which were out of scope of this paper. This shows that a significant portion of the application could be generated directly from the use cases and their scenarios.

The application logic code (e.g. the “selectsOK” method) presented in Figure 18 can be generated using the transformation rules introduced in this paper. However, the

domain logic code (e.g. method “validates” in CStudent). Thus, we need a mechanism to determine code which needs rework. This can be done by examining code in Figures 12 and 18. The difference is quite obvious and the additional methods can be easily indicated. Also, the methods that can be retained are easily detected.

4. Conclusion and future work

The most important aim of this work was to demonstrate that general-purpose requirements models can have operational (runtime) semantics and this semantics can be used to achieve high levels of automation in code evolution. As the presented case study shows, our approach automates to significant extent the transition from requirements written in constrained natural language to code. We have demonstrated that the presented semantics allows for generating Java code within the application logic layer.

It is important to note that the presented RSL language fulfills the requirements postulated in [22]. Its understandability to the users was validated in the industrial context and the results presented in [23]. In this paper we have shown that RSL is also precise enough to control code generation. Thus we can claim that the new approach, together with the ReDSeeDS tool, constitute a powerful mean of support for the software developers. The main benefit is that the developers can quickly see the consequences of changes to application logic on the domain logic. This ‘diff’ can be determined for even a yet not existing code right after the requirements are ready. The differences are illustrated by instantly showing places in the domain-specific code where code rework is needed. Another important benefit is that the system refactors the code automatically based on the changed scenarios and domain notions. The developers have clear pointers, as to which methods should be changed (extended, combined, split, etc.). This is clearly shown by comparing visually with the previous version.

A software development organisation can base its software lifecycle around a repository of RSL-based requirements models. Within this repository, different variants of the same system (produced in different iterations) can be stored. The repository can also contain models for other systems, produced in different domains. It can be noted that after conducting several projects, a development organisation obtains a powerful repository of reusable artifacts. These artifacts can be reused by building new versions, but also a wider (cross-domain, cross-system) reuse is possible. This can be done by merging use cases and associated code.

Requirements written in RSL offer specific solutions to specific problems. However, it can be noted that RSL models can be also treated as certain application logic patterns. If we detach the application logic from the specific domain we obtain a generalised model which can be treated as a pattern for visible functionality of software (see. also Figure 2). This seems to be an interesting research direction and has been recently elaborated in [24]. Another interesting direction is to introduce into RSL certain constructs that would allow for specifying domain-specific elements in more detail. By combining with the pattern approach, this direction would allow for constructing applications based on pre-defined domain specifications and associated patterns of application-logic functionality. The new domain-specific element of RSL could include an algorithmic language for specifying domain logic processing (as opposed to application logic processing offered already by RSL). Also, certain notation could be introduced for repre-

senting domain-specific data objects (structure of the domain). This could be combined with possible integration of RSL with the existing domain-specific languages. Domain-specific languages could be plugged-in at the domain logic level and substitute the above mentioned domain-specific elements of RSL.

References

- [1] Miller, J., Mukerji, J., eds.: MDA Guide v. 1.0.1, omg/03-06-01. Object Management Group (2003)
- [2] Stahl, T., Völter, M.: Model Driven Software Development: Technology, Engineering, Management. Wiley (2006)
- [3] Kelly, S., Tolvanen, J.P.: Domain Specific Modeling: Enabling Full Code Generation. Wiley (2008)
- [4] Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison Wesley Professional (2002)
- [5] Greenfield, J., Short, K.: Software Factories. Assembling Applications with Patterns, Models, Frameworks and Tools. Wiley, Indianapolis, Indiana (2004)
- [6] Mens, T., Demeyer, S., eds.: Software Evolution. Springer Verlag (2008)
- [7] Ernst, N.A., Mylopoulos, J., Wang, Y.: Requirements evolution and what (research) to do about it. Lecture Notes in Business Information Processing **14** (2009) 186–214
- [8] Nuseibeh, B.: Weaving together requirements and architectures. IEEE Computer **34**(3) (2001) 115–117
- [9] Sutcliffe, A.: On the inevitable intertwining of requirements and architecture. Lecture Notes in Business Information Processing **14** (2009) 168–185
- [10] Barber, K.S., Graser, T.J., Grisham, P.S., Jernigan, S.R.: Requirements evolution and reuse using the systems engineering process activities (sepa). Australian Journal of Information Systems **7**(1) (1999) 75–97 Special Issue on Requirements Engineering.
- [11] Zowghi, D., Gervasi, V.: On the interplay between consistency, completeness, and correctness in requirements evolution. Information and Software Technology **45** (2003) 993–1009
- [12] Moon, M., Yeom, K., Chae, H.S.: An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. IEEE Transactions on Software Engineering **31**(7) (2005) 551–569
- [13] Śmiałek, M.: From user stories to code in one day? Lecture Notes in Computer Science **3556** (2005) 38–47 XP 2005.
- [14] de Boer, R.C., van Vliet, H.: On the similarity between requirements and architecture. Journal of Systems and Software **82**(3) (2009) 544 – 550
- [15] Du Bois, B., Demeyer, S.: Accommodating changing requirements with EJB. Lecture Notes in Computer Science **2817** (2003) 152–163
- [16] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Reading (1992)
- [17] Object Management Group: Unified Modeling Language: Superstructure, version 2.2, formal/2009-02-02. (2009)
- [18] van den Berg, K.G., Simons, A.J.H.: Control flow semantics of use cases in UML. Information and Software Technology **41**(10) (1999) 651–659
- [19] Śmiałek, M., Bojarski, J., Nowakowski, W., Ambroziewicz, A., Straszak, T.: Complementary use case scenario representations based on domain vocabularies. Lecture Notes in Computer Science **4735** (2007) 544–558
- [20] Kaindl, H., Śmiałek, M., Svetinovic, D., Ambroziewicz, A., Bojarski, J., Nowakowski, W., Straszak, T., Schwarz, H., Bildhauer, D., Brogan, J.P., Mukasa, K.S., Wolter, K., Krebs, T.: Requirements specification language definition. Project Deliverable D2.4.1, ReDSeeDS Project (2007) www.redseeds.eu.
- [21] Śmiałek, M., Bojarski, J., Nowakowski, W., Straszak, T.: Scenario construction tool based on extended UML metamodel. Lecture Notes in Computer Science **3713** (2005) 414–429
- [22] Śmiałek, M.: Accommodating informality with necessary precision in use case scenarios. Journal of Object Technology **4**(6) (2005) 59–67
- [23] Mukasa, K.S., Jedlitschka, A., Graf, C., et al.: Requirements specification language validation report. Project Deliverable D2.5.1, ReDSeeDS Project (2007)
- [24] Ambroziewicz, A., Śmiałek, M.: Application Logic Patterns - Reusable Elements of User-System Interaction Lecture Notes in Computer Science **6394** (2010) 241–255

This page intentionally left blank

Tools, Techniques and Languages for Model Driven Development

This page intentionally left blank

An Approach for Enacting Software Development Process: SPEM4MDA

Vladimirs NIKULSINS¹, Oksana NIKIFOROVA and Jurijs KORNIJENKO
Riga Technical University, Institute of Applied Computer Systems

Abstract. A promising approach, Object Management Group's (OMG) Model Driven Architecture (MDA), shifts the focus of software development from writing code to building models. It gives no guidelines on software implementation in terms of activities, phases, roles and responsibilities. Authors of this paper are offering an approach for enacting software development process with the help of process models. They are formalized with the help of Software Process Engineering Metamodel (SPEM), therefore the approach is called SPEM4MDA. The essence of approach is to use existing non-MDA process model formalized in SPEM notation, and with a help of another OMG initiative – Query/View/Transformation (QVT) – transforming it to MDA compliant process model.

Keywords. MDA, QVT, SPEM, software development process, model transformations

Introduction

Modern software development process frameworks are making distinction between the method content and process definitions. An example here could be Rational Unified Process (RUP), which is aligning software development process into both static aspects like disciplines and tasks, and also dynamic aspects like phases and iterations which are involved in the software development process [1].

In the same way SPEM metamodel is organized: method content is expressed using work product definitions, role definitions, task definitions, and guidance. Process is expressed with activities, which can be nested to define breakdown structures as well as related to each other to define dynamics of a process. Both method content and process are linked with guidance, which provides linkage between these two concepts, and is usually expressed in guidelines, whitepapers etc. [2].

Authors of this paper previously concentrated their attention on analyzing processes and process related artifacts for model-driven development, and also providing some solutions to support software development process transition from “traditional” software development process to the MDA compliant process using process models. The model-driven approach itself and its principles are used to transform “traditional” software development model to the MDA compliant in proposed SPEM4MDA approach. The source model presenting “traditional” software development process is transformed into the target model, which presents the MDA-

¹ Corresponding author: Vladimirs Nikulsins, Riga Technical University, Meza 1/3, LV-1048 Riga, Latvia; E-mail: ranva@inbox.lv

based software development process. Query/View/Transformation (QVT) is used to support such transition between models.

The practical validation of such approach was also performed in one of the Latvian IT companies. The goal of such validation was to provide a model of model-driven development and analyze feasibility of the MDA application within this company.

The structure of this paper is organized as follows. Current section provides general introduction to the problem area. Section 1-2 consists from problem statement and related works. Then the main chapter starts, which clarifies the approach proposed: “3. Essence of SPEM4MDA Approach”. This chapter is divided into the following sub-sections: “3.1. Knowledge Flow to Develop SPEM4MDA”, “3.2. High Level Sequence of Steps to Implement SPEM4MDA” and “3.3. Process Modelling Tools”. Section 4 briefly describes practical validation aspects, and section 5 concludes the work.

1. Problem Statement

The essential criterion of success in software development lies in both selection and utilization of efficient software development approach. There is known problem on how to apply model-driven software development approach within the organization, which is using so called “traditional” software development approach. The knowledge on how to deploy model-driven development practices is not well formalized, and in order to help companies implement model-driven approach, help from experienced consultants is required [3]. Technology and process issues are described more in the next section.

This paper tries to resolve formalization issue and provide an approach which could be used in various software development companies to adopt model-driven development. Practical validation was also a part of this research.

Though, there are also other issues, related to model-driven development approach. When thinking about process modelling, additional aspects which should be taken into account are:

- Lack of methodological guidelines (how to use a modelling tool in real project)
- Transformation chain is not complete (tool interoperability issue) [4].
- Problem of tool integration and export/import of model (usually in XML). Even when some tools provide real-time interfaces, they are actually quite limited in practical usage. More information can be found from [5].

2. Related Works

This section contains overview of currently existing solutions and approaches for model-driven transformations which can be utilized for SPEM model transformations. The authors of this paper already concentrated their efforts on MDA integration into the traditional software development lifecycle from methodological point of view, as well as reviewed Software Development Life Cycle (SDLC) aspects [6], [7], [8] and [5]. Software development activities, workflows and software development team organization were also analyzed in the previous researches [9], [10].

Additional information on MDA terminology and abbreviations can be found from [3].

SPEM [2] is widely used in practice. It is de-facto standard, which can be readily applied to modelling software life cycle processes. SPEM can be examined in relation to project management and software life cycle processes definition. It can be supplemented by a range of formal standards that include the CMMI [11], Rational Unified Process [1], ISO [12] and IEEE [13] standards. Various researches and case studies are available on this area. The most significant ones from the author's point of view are described below.

In order to apply the rules of model, meta-model and meta-meta-model definition, SPEM as meta-model of process definition can be enriched with additional components at the meta-meta-model level. The authors of [14] underline that all process meta-models are built on a core set of concepts, which are always more or less the same (work items, products and resources). In addition to these common modelling constructs they integrate domain related concepts, which are specifically dedicated to the needs of expression of the user. With the standardization of meta-meta-model all these meta-models may be described using the same formalism [14].

[15] suggests enhancing SPEM with OCL [16], because it lacks a formal description of its semantics that makes it hard to use. Additional problem is that using SPEM is difficult because the OMG proposal is very generalist and provides no directives on how to use it. In future it is planned to define operational semantic for SPEM refined with OCL, not only structural.

[17] investigates different approaches to model transformations. Particularly, its attention is paid to OMG's Queries/Views/Transformations (QVT) [18]. [17] reviews problems of tracing activities and models conformance, when the changes are made on the both sides – source model and target model. Unification of declarative and imperative issues on model transformations is proposed by adopting new approach to model transformations. It is capable of providing a flexible, efficient, and practical platform for creating model transformations, which can be re-used in relation to SPEM. Some related work where SPEM was transformed to BPMN through the workflow automation is reviewed in [19]. General Standard Software Process is applied to MDA in SPEM within MASTER project [20]. It provides standard workflows and activity model using SPEM 1.0. Therefore SPEM can be utilized as a general meta-model for defining processes. Model-to-model transformations can be applied to SPEM models, like they are applied to business process models according to MDA. But in the early versions of SPEM there was a lack of formalization and layering was not supported.

The QVT Relations language was suggested by authors of this paper for SPEM metamodel transformations in [21].

The process meta-models integration and unification problem is reviewed in [15]. This paper named "Process-Centered Model Engineering" was written in 2001, however, since this time the problem became even more serious because of the different MOF implementations.

3. Essence of SPEM4MDA Approach

This chapter describes the information sources used to develop SPEM4MDA, and also the architecture and steps to be taken to implement this solution. When planning "traditional" software development process transition to model-driven, the first thing to

start with is current software development process analysis. The help from some experienced MDA process architect consultants might be required. One of the OMG initiatives for smooth transition to MDA is special program “MDA FastStart”. It is divided into three main phases – assessment, overview and transition [14].

However, these actions are mainly based on the expertise of consultants, and are not formalized into some specific framework or methodology. With the approach described in this paper, it is possible to formalize some part of the transition process, which covers software development description and transition to MDA process through the process model.

3.1. Knowledge Flow to Develop SPEM4MDA

In the previous author works, software development process (SWEBOK), standards (ISO, CMMI) and methodologies (RUP, MSF, XP) were investigated. Model-Driven Development was also analyzed and decomposed into static and dynamical representation, as well as linked from methodological point of view to RUP, MSF and general guidelines for any software development process [6], [7]. Process static and dynamical representation consists from two objects.

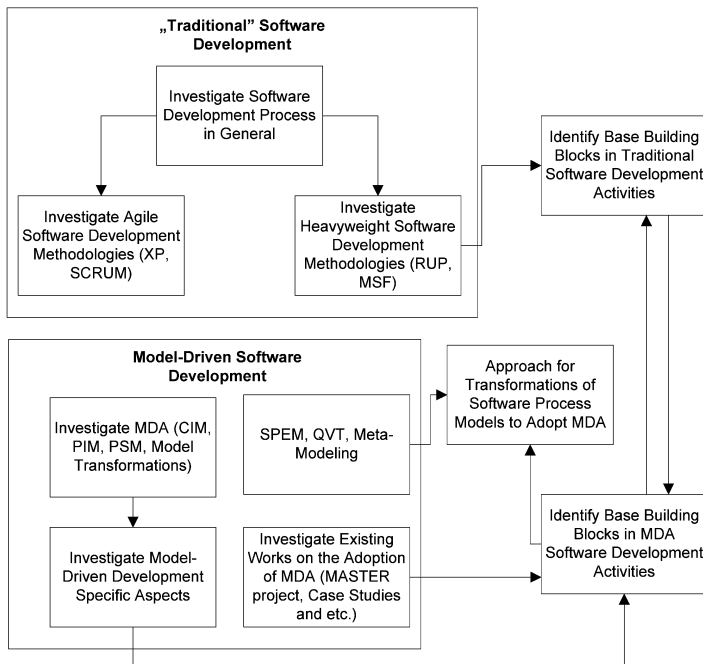


Figure 1. Graphical representation of knowledge flow to develop SPEM4MDA solution

Additional sources of information were investigated from various researches and case studies on model-driven development and its specific aspects utilization. As a result it lead to the gathering of information for both “traditional” software development and model-driven.

Such process model element decomposition allows linking activities from “traditional” processes to model-driven (for example, PIM modelling activity instead of coding). All elements of process model can be formalized into chosen process model notation, and decomposed with the help of base building blocks (entities, which exist in every software development process). OMG standard for modelling software life cycle processes is SPEM. The process models should be expressed using SPEM base building blocks. In order to link them, the usage of special language is required. Linking base building blocks means that it is possible to transform one model into another: from “traditional” software development process model into MDA. QVT is used to achieve this goal.

As a result, it is possible to create a process model of “traditional” software development process in SPEM, execute QVT transformation and get an MDA based software development process model.

The information flow and graphical representation of the approach presented is depicted on Figure 1.

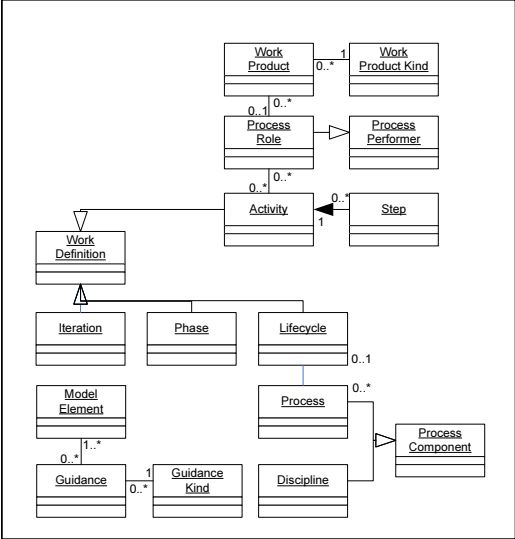
3.2. High Level Sequence of Steps to Implement SPEM4MDA

Steps on information gathering and analysis described in the previous chapter are omitted. High-level approach consists from the following steps:

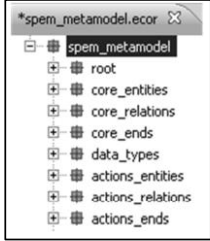
1. Create/adapt SPEM metamodel. All SPEM models used in transformations must conform to single metamodel. This is also the base for writing QVT rules. Adaption is required to handle multiplicity in elements, which will be further processed with QVT (e.g., the same activity can be named differently, therefore some standardization is required).
2. Importing metamodel to *.ecore format and validation. This step ties the SPEM metamodel to Eclipse platform, and is tightly related to the first one.
3. Create base building blocks for traditional Software Development Process (SDP). Usage of base building blocks in the process modelling allows avoiding problem of unified terminology, since the same activities may be named differently, and also different process architects can produce different models. Similar approach is implemented in the process modelling tools like Eclipse EPF [22].
4. Define QVT Relations transformations for SPEM elements converted in a model-driven context. Transformations will modify the current model without changing non-standard elements (e.g., there might be some company specific processes which are not defined as building blocks). QVT transformation provided on Figure 2 copies source activity to target, changing the activity name from “Provide a demo of program solution” to “Provide an executable model of program solution”. Both source model *clientprocess* and target model *mdd* belongs to the same metamodel SPEM, which is expressed in Ecore format. Additional constraint is that the transformation occurs only for Inception phase.
5. Create source process model in SPEM. This step involves modelling activities for some real software development process. Base building blocks from the previous step are used as a source.
6. Execute QVT transformation (target model, MDA specific). The transformation rules from step 4 are executed, producing an output model which conforms to MDA development process.

The steps above are graphically represented on Figure 2. Steps 5 and 6 are target of interest for end users, while steps 1 to 4 are describing the sequence of steps when building SPEM4MDA solution as such.

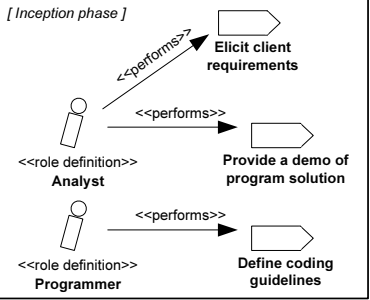
Step 1. Create/adapt SPEM metamodel



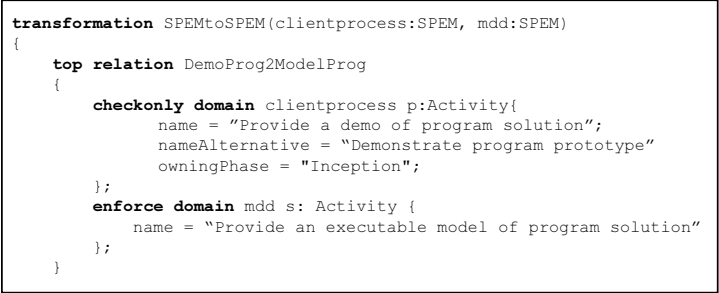
Step 2. Import metamodel to *.ecore format and validate it



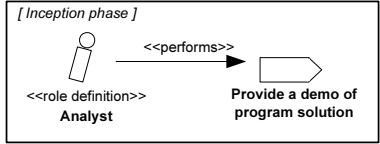
Step 3. Create base building blocks for traditional SDP



Step 4. Define QVT transformations



Step 5. Create source process model in SPEM



Step 6. Execute QVT transformation (target model, MDA specific)

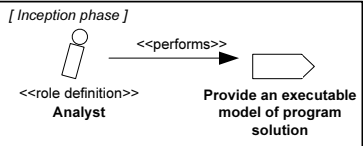


Figure 2. Sequence of steps when implementing SPEM4MDA

Solution of applying SPEM transformations based on the pre-defined QVT rules allows partly automating the work of process engineer and supporting him with the model-driven development lifecycle utilization.

3.3. *Process Modelling Tools*

MDA is aimed to decrease the dependency on the platform proposing platform independent models as the most important artefact. But the paradox is that currently there is a huge amount of modelling software, which is not using shared standards for storing models. As a result, the models created in one tool might be totally useless in another tool due to the lack of integration between them. This problem affects SPEM as well.

There are various tools, which support SPEM notation in the software development life cycle modelling. One of most widely known is IBM Rational Architect, where SPEM is used as a core notation for representing the process workflow. It is possible to transform SPEM model into the different information sources. E.g., align tasks from SPEM in Microsoft Project, or save its structure in XML [11]. SPEM notation is used as a main notation for RUP activities and processes representation.

Objectteering SPEM Modeler is a BPMN graphical editor, which supports complete UML2 standard modelling and BPMN modelling for business processes integrated with UML. It also provides various transformation possibilities [23].

MagicDraw UML 16.0 (with SPEM extension). This tool is fully UML 2.0 compliant. The tool supports UML 2.0 standard, code engineering for multiple programming languages (Java, C++, C# and others) as well as for data modelling. The tool has teamwork facilities and supports integration with the various IDEs. MagicDraw has integration with various MDA tools: Compuware OptimalJ, AndromDA, Interactive Objects' ArcStyler and etc. [24]. There are two add-ons available, which are the most important in relation to this research. These are methodology wizard and SPEM plug-in.

The main process modelling tool in Eclipse, which is used as a platform for the proposed solution, is called EPF Composer. It is a tool, aimed to produce a customizable software process engineering framework. Once the model is defined in the EPF, it can be exported to the external file in XML format. The Eclipse package in which EPF Composer models are stored is called UMA.

SPEM4MDA model transformations are performed using mediniQVT tool, which is able to operate with the models expressed as Ecore metamodels and provides debugging features and transformation rules tracing. The QVT Relations transformations are applied to the source model, which corresponds to the "traditional" software development lifecycle. mediniQVT uses Eclipse platform as well.

4. **Practical Validation**

The practical validation of SPEM4MDA approach was performed in one of the Latvian IT companies, delivering software solutions for travel agencies. It is developing solutions based on Lotus platform.

During interviews with customer, custom Software Development Life Cycle (SDLC) notation was used to represent the software development process in the

company. Modeling directly with SPEM base building blocks might save some time and resources, however in this particular case usage of custom SDLC was chosen with special intent to make a transition from some custom notation to SPEM building blocks (Figure 3).

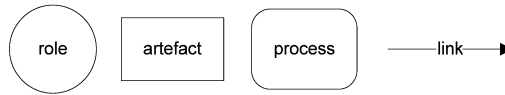


Figure 3. Custom SDLC elements

We did an analysis of the software development lifecycle, and identified 5 phases:

- Analysis
- Design
- Development and Test
- Stabilize
- Deploy

A fragment from Design phase used in company is depicted on Figure 4. The company suggests adapting some standard solution to the client needs, similar to Enterprise Resource Planning (ERP) solutions implementation. Thus, clients can both acquire new technological solution and also enhance their business processes.

Since the life cycle for company is described using custom notation and terms, there is a need for unification and usage of common activities and terms. According to the SPEM4MDA approach proposed, SDLC model should be expressed using SPEM and base building blocks (in order to link standard activities with model-driven activities).

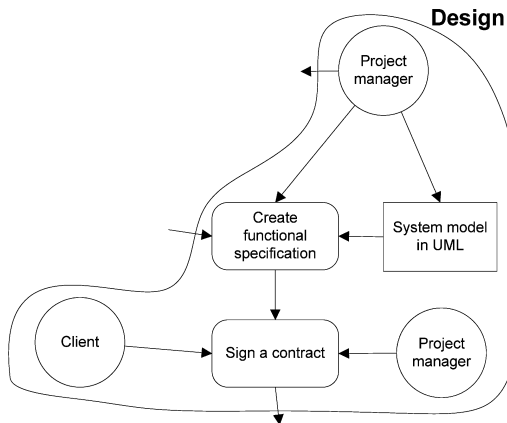


Figure 4. A fragment from Design phase in custom SDLC notation

When Company's development process was analyzed, there was a need to define enrichment activities for every development phase. General guidelines on how to enrich "traditional" software development methodologies with model-driven specific can be found from [5], [6], [7].

According to the proposed approach, transformation rules were defined with the help of QVT (Query/View/Transformation) standard. First, we were building SPEM model which represents traditional software development. It might be composed both from Open UP base building blocks (such building block can include activities, their relations, processes, milestones, roles and etc.), and custom building blocks (e.g., some organization-specific activities). All of them should be linked together in one model. SPEM model should be exported to the Ecore model. The QVT plug-in mediniQVT works with Ecore models, so this is mandatory requirement. Once the source model is loaded into mediniQVT, transformation should be initialized. A result of transformation is SPEM model in Ecore, which represents model-driven software development life cycle for organization.

Several transformations occurred automatically, other ones manually. For simple SPEM models with limited number of elements like it was in this practical case, it is not mandatory to use QVT tools and transform models automatically. Transformations can be done manually, using any SPEM modeling tool. But from the other side, if any complex rules or OCL constraints are defined for the model elements, this requirement becomes mandatory. Additionally, if the input model is created from SPEM base building blocks, it is just 'one-click' activity to make the transformation and receive target model. Additional information and examples of SPEM models can be found from [25].

5. Conclusions

It is possible to represent any software development life cycle using SPEM. However, any modeling activities related to the system being investigated, are pretty subjective and dependant on the expert who is responsible for modeling. Therefore the use of appropriate SPEM base building blocks is required. SPEM is OMG standardized notation and is successfully used for various types of lifecycles. Using process model it is possible to get the MDA compliant model by transforming source SPEM model with QVT.

When authors of this paper were trying to develop model of existing so called "traditional" software development life cycle, the most essential problem was the difference between detail levels. In our case it gives the general understanding of the software development process, but at the same time might be not enough detailed to link the SPEM model with base building blocks.

There are following restrictions identified: the model-driven process model obtained from the initial model cannot replace "action plan" for the manager. It also says nothing regarding specific MDA transformation tools and chains, since there should be different reasoning on the process of MDA toolset selection.

Usage of MDA process is reasonable only in long-term, since if Company will try to change their current process immediately, they will need to consider training costs, re-building existing assets, buying new licenses for tools, changing the way they support their clients (due to usage of different development paradigm), and etc.

Also, MDA is preferred solution when there is a need to change a platform. The Company which was analyzed is dependent on IBM Lotus client-server platform. If there will be need to move to a newer platform or provide different kinds of service, MDA implementation might be an option.

Changes in team requires hiring experienced system architect into the team, which will enforce the knowledge exchange between the team and will be able to control if the approach is compliant with MDA or not. Model-driven development is very strict when it comes to the process management. The programmers should not follow code-oriented thinking when changes to the system are needed. It is supposed that they will work with models.

This research is limited to the definition of the process model and executing transformations only, thus formalizing only part of process architect knowledge. It can't fully replace human expertise.

References

- [1] Kruchten, P.: The Rational Unified Process An Introduction, Second Edition, Addison Wesley (2000)
- [2] OMG Software & Systems Process Engineering Metamodel specification (SPEM), version 2.0, <http://www.omg.org/spec/SPEM/2.0> (2008)
- [3] OMG Model Driven Architecture, <http://www.omg.org/mda>
- [4] Nikiforova O., Cernickins A., Pavlova N.: On the Tool Chain for Model Driven Architecture and Model Driven Development: Drawing a Distinction. The 13th East-European Conference on Advances in Databases and Information Systems (ADBIS), September 2009. – Riga, Latvia: JUMI Publishing House Ltd., pp. 408--415 (2009)
- [5] Nikulsins, V., Nikiforova, O.: Tool Integration to Support SPEM Model Transformations in Eclipse. The 50th International Scientific Conference of Riga Technical University. RTU (2009)
- [6] Nikiforova O., Nikulsins V., Sukovskis U.: Integration of MDA Framework into the Model of Traditional Software Development, In the series "Frontiers in Artificial Intelligence and Applications", Databases and Information Systems V, Selected Papers from the Eighth International Baltic Conference Baltic DB&IS 2008, Haav H.-M., Kalja A. (Eds.), IOS Press, pp. 229--242 (2009)
- [7] Nikulsins, V., Nikiforova, O., Sukovskis, U.: Mapping of MDA Models into the Software Development Process, Databases and Information Systems, Proceedings of the Eighth International Baltic Conference Baltic DB&IS 2008, H.-M. Haav and A. Kalja (Eds.), Tallinn University of Technology Press, Tallinn, Estonia, June 2-5, pp. 217--226 (2008)
- [8] Nikulsins V., Nikiforova O.: Adapting Software Development Process towards the Model Driven Architecture, Proceedings of The Third International Conference on Software Engineering Advances (ICSEA), International Workshop on Enterprise Information Systems (ENTISY), 2008, Mannaert H., Dini C., Ohta T., Pellerin R. (Eds.), Sliema, Malta, October 26-31, 2008., Published by IEEE Computer Society, Conference Proceedings Services (CPS), pp. 394--399 (2008)
- [9] Nikulsins, V., Nikiforova O., Sukovskis U.: „Analysis of Activities Covered by Software Engineering Discipline”, Databases and Information Systems, Seventh International Baltic Conference on Databases and Information Systems, Communications, Materials of Doctoral Consortium, O. Vasilecas, J. Eder, A. Caplinskas (Eds.), pp. 130--138, VGTU Press „Technika” scientific book No 1290, Vilnius, Lithuania (2006)
- [10] Nikulsins, V., Nikiforova, O.: “Software Development Teams Organization”, The 46th Scientific Conference of Riga Technical University, Computer Science, Applied Computer Systems, October 13-14, Riga, Latvia, 2005, published in the 5th Series “Computer Science. Applied Computer Systems, Vol. 26, pp. 54--65 (2006)
- [11] Software Engineering Institute. Capability Maturity Model Integration (CMMI), <http://www.sei.cmu.edu/cmmi/>
- [12] ISO - International Organization for Standardization, <http://www.iso.org/>
- [13] IEEE Standards Association, <http://standards.ieee.org>
- [14] Breton, E., Bezivin, J.: Process-Centered Model Engineering. France (2001), <http://www.sciences.univ-nantes.fr/lina/atl/www/papers/edoc.pdf>
- [15] Combemale, B., Cregut, X.: Towards a Rigorous Process Modeling With SPEM. France (2006), <http://www.combemale.net/research/phd/2006/iceis250406-CCCC-poster401.pdf>
- [16] Object Constraint Language Specification, version 2.0 (2006), <http://www.omg.org/cgi-bin/apps/doc?formal/06-05-01.pdf>
- [17] Tratt, L.: Model transformations and tool integration. Department of Computer Science, King's College London, Springer-Verlag (2004)

- [18] Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.0 (2008), <http://www.omg.org/docs/formal/08-04-03.pdf>
- [19] Debnath, N., Zorzan, F.A., Montejano, G., Riesco, D.: Transformation of BPMN subprocesses based in SPEM using QVT. *Electro/Information Technology, 2007 IEEE International Conference*. pp. 146--151 (2007)
- [20] Model-driven Architecture inSTRumentation, Enhancement and Refinement. Process Model to Engineer and Manage the MDA Approach. European Software Institute (2003), <http://modeldrivenarchitecture.esi.es/pdf/Deliverable-D32.zip>
- [21] Nikulsins, V., Nikiforova, O.: Transformations of SPEM Models Using Query/View/Transformation Language to Support Adoption of Model-driven Software Development Lifecycle. *The 13th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, September 2009. Riga, Latvia, JUMI Publishing House Ltd. (2009).
- [22] Eclipse Process Framework (EPF). Prepared by Eclipse Foundation: Eclipse – <http://www.eclipse.org/epf>. (2009)
- [23] Objectteering, http://www.objectteering.com/products_uml_modeler.php
- [24] MagicDraw UML, <http://www.magicdraw.com>
- [25] Nikulsins V., Nikiforova O., Kornijenko J. SPEM model transformations to adopt MDA in practice. *Databases and Information Systems. Proceedings of the Ninth International Baltic Conference Baltic DB&IS 2010. Databases and Information Systems, 2010*. Barzdins J., Kirikova M. (Eds.). 2010. July, Latvia, Riga. Riga: University of Latvia Press, Riga, Latvia, pp. 295-307 (2010)

Bringing Domain Knowledge to Pattern Matching

Agris SOSTAKS

IMCS University of Latvia, Latvia, e-mail: agris.sostaks@lumii.lv

Abstract. This paper addresses the pattern matching problem for model transformation languages. Despite being an NP-complete problem, the pattern matching can be solved efficiently in typical areas of application. Prediction of actual cardinalities of model elements is the key to sufficient efficiency. The existing approaches acquire the actual cardinalities using complex run-time model analysis or using analysis of metamodel where the required information is poorly supplied. In the paper we show how the deeper understanding of domain which is targeted by model transformation language can dramatically reduce the complexity of pattern matching implementation. We propose a simple pattern matching algorithm for model transformation MOLA which is efficient for tasks related to the model driven software development. Additionally a metamodel annotation mechanism is proposed. It refines the existing means of metamodeling by adding new classes of cardinalities. They make more efficient the pattern matching algorithms which do not use the complex run-time analysis.

Keywords. model transformation, MOLA, domain-specific, pattern matching, annotations

Introduction

Model transformation languages are becoming increasingly mature in recent years and range of the areas where transformation languages are being used is widening. The growing popularity of transformation languages puts stricter requirements on their efficiency. Most of the popular transformation languages are using declarative pattern definition constructs. The main implementation problem of such languages is the pattern matching. This problem, in fact, is the subgraph isomorphism problem which is known to be NP-complete. However, in practice typical patterns can be matched efficiently using relatively simple methods. The use of different means of pattern definition results into different implementations of pattern matching for every language. The more sophisticated constructs a language use, the more complicated becomes the implementation of the pattern matching. The pattern matching implementation for the graphical model transformation language MOLA [1] is addressed by this paper.

One of the most popular application domains for model transformations is Model Driven Software Development (MDSD) related tasks. These tasks have been tried to be solved in almost every model transformation language, also in MOLA. In fact, MOLA is designed as a simple and easy readable (therefore graphic!) model transformation language, which would cover typical transformation applications in MDSD. We refer to

transformations used in the European IST 6th framework project ReDSeeDS¹ to illustrate the typical MDSD use cases of MOLA. We present a brief overview of MOLA in Section 1.

Section 2 gives a survey of pattern matching strategies used in implementations of model transformation languages. One of the most popular and also the most efficient method to solve the pattern matching problem is the local search plan generation. This method is used by several implementations of transformation languages, e.g. PROGRES [2], VIATRA [3], GrGen [4] and Fujaba [5]. However, each implementation varies in details depending on pattern definition constructs used in the language. Most sophisticated strategies even use statistical analysis of model in runtime.

The implementation of pattern matching in MOLA is discussed in Section 3. MOLA also uses the local search plan generation strategies for pattern matching. We analyse the main properties of tasks related to MDSD. The typical patterns appearing in ReDSeeDS project have been discovered. The research is a base for a simple heuristic algorithm that is efficient for most patterns typically used in MDSD-related transformations. This algorithm can be used in fast implementations of pattern matching for transformation languages which use similar constructs to MOLA. The key aspect to success of the simple algorithm is finding out the actual cardinalities which are specific to the MDSD domain. The development of the simple pattern matching algorithm has revealed that metamodelling languages do not offer sufficient means to denote the actual cardinalities. Therefore we introduce two new classes of cardinalities and a metamodel annotation mechanism which allows specifying the refined cardinalities in metamodel. In Section 3 we also show how the new annotation mechanism can be used to improve the efficiency of pattern matching algorithm.

1. MOLA

MOLA is a graphical transformation language developed at University of Latvia, Institute of Computer Science and Mathematics. It is based on a traditional concept among transformation languages: the pattern matching. The main distinguishing feature is the use of simple procedural control structures governing the order in which pattern matching rules are applied to the model. The formal description of MOLA and also MOLA tool can be found in MOLA web site².

One of the biggest use cases of MOLA is in the European IST 6th framework project ReDSeeDS. One of the goals in ReDSeeDS is providing a tool support for MDSD. The Software Case Language (SCL) is used to model the system. The main parts of SCL are Requirements Specification Language (RSL)[6] and a subset of Unified Modelling Language (UML). Requirements in RSL are scenarios in a controlled natural language. MOLA is used to specify a transformation from requirements to architecture and further to detailed design models. Transformations are divided into several steps generating a chain of models. We have gained a great experience during this project in writing typical MDSD transformations. Short excerpts from these transformations are used in this paper.

A MOLA program transforms an instance of a source metamodel (the source model) into an instance of a target metamodel (the target model). Source and target metamodels

¹<http://www.redseeds.eu>

²<http://mola.mii.lv>

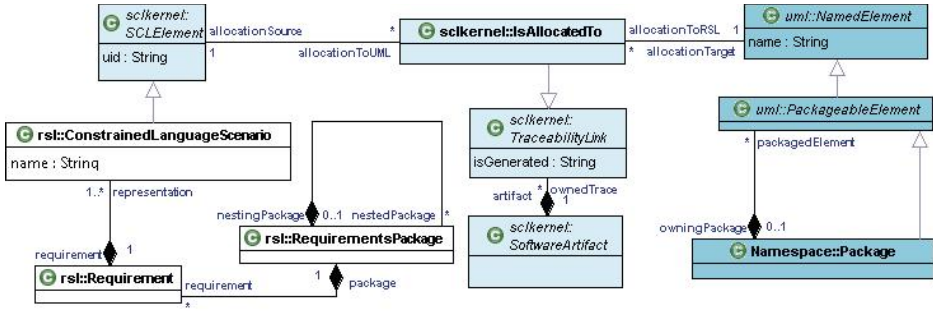


Figure 1. Fragment of SCL used in transformation example

are jointly defined in the MOLA metamodeling language, which is quite close to the OMG EMOF specification. Actually, the division into the source and target parts of the metamodel is quite semantic, they are not separated syntactically (the complete metamodel may be used in transformation procedures in a uniform way). Typically, additional traceability associations or classes link the corresponding classes from source and target metamodels; they facilitate the building of natural transformation procedures and document the performed transformations.

Figure 1 demonstrates part of the SCL metamodel used in this section as a transformation example. It is a simplified version of the full SCL metamodel. As we can see the source and target metamodels are given in the same class diagram and the separation into source (RSL) and target (UML) metamodels is quite semantic. The requirements specification (requirement model) consists of requirements packages which are used to group the requirements in RSL. One of the requirement representation forms is the constrained language scenario. Thus a requirement may consist of several scenarios written in the constrained language. The traceability elements also play a significant role in MOLA transformations. The *sclkernel::TraceabilityLink* class (particularly, the *IsAllocatedTo* for mapping from requirements to architecture) is used for this purpose.

MOLA procedures form the executable part of a MOLA transformation. One of these procedures is the main one, which starts the transformation. MOLA procedure is built as a traditional structured program, but in a graphical form. Similarly to UML activity diagrams, control flows determine the order of execution. Call statements are used to invoke sub-procedures. However, the basic language elements of MOLA procedures are specific to the model transformation domain - they are rules and loops based on rules. Rules embody the declarative pattern match paradigm, which is typical to model transformation languages.

Each rule in MOLA has the pattern and action part. Both are defined by means of class elements and links. A class element is a metamodel class, prefixed by the element ("role") name (graphically shown in a way similar to UML instance). An association link connecting two class elements corresponds to an association linking the respective classes in the metamodel. A pattern is a set of class elements and links, which are compatible to the metamodel for this transformation. A pattern may simply be a metamodel fragment, but a more complicated situation is also possible - several class elements may reference the same metamodel class. A class element may be a reference to previously matched instance. This reuse mechanism plays a crucial role in the implementation of

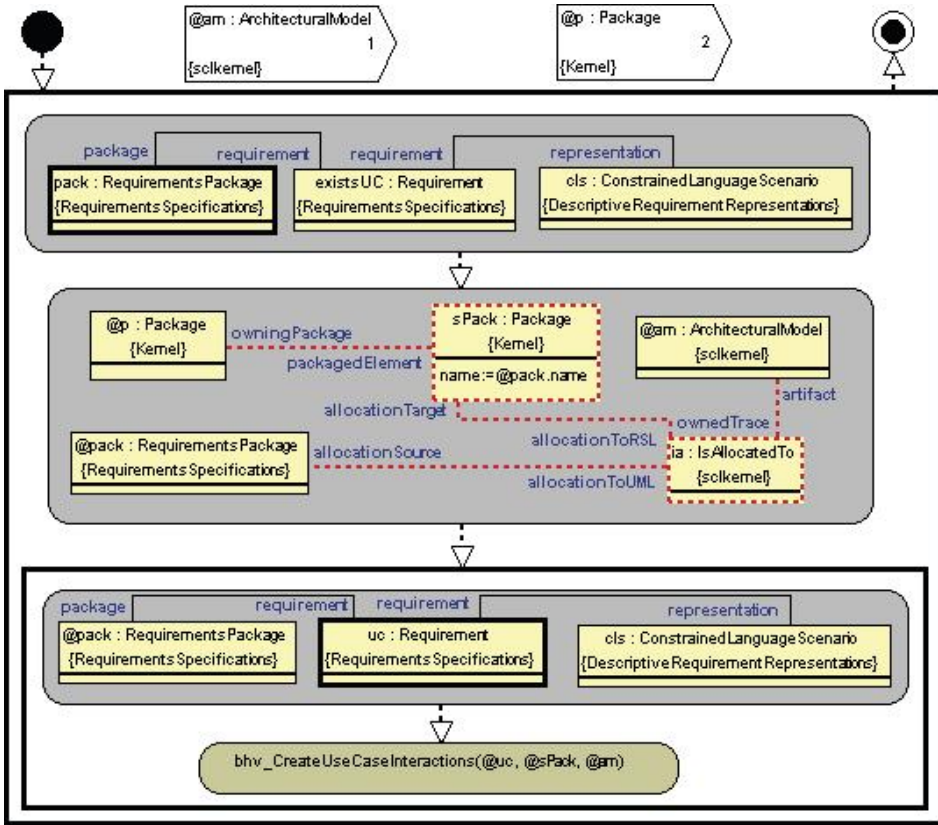


Figure 2. Transformation example - MOLA procedure building package hierarchy

pattern matching. In addition, a class element may contain also a constraint - a simple Boolean expression. The main semantics of a rule is in its pattern match - an instance set in the model must be found, where an instance of the appropriate class is allocated to each pattern element so that all required links are present in this set and all constraints evaluate to true. If such a match is found, the action part of the rule is executed. The action part also consists of class elements and links, but typically these are creation actions. Instances may also be deleted and modified in the action part. If several instance sets in the model satisfy the rule pattern, the rule is executed only once on an arbitrarily chosen match. Thus, a rule in MOLA typically is used to locate some fragment in the source model and build a required equivalent construct in the target model.

Another essential construct in MOLA is the foreach loop. The foreach loop is a rectangular frame, which contains one special rule - the loophead. The loophead is a rule, which contains one marked (by a bold border) class element - the loop variable. A foreach loop is an iterator, which iterates through all instances of the loop variable class, which satisfy the constraint imposed by the pattern in the loophead. With respect to other elements of the pattern in the loop head, the "existential semantics" is in use - there must be a match for these elements, but it does not matter, whether there is one or several such matches. A loop typically contains the loop body - other rules and nested loops, whose

execution order is organised by control flows. The loop body is executed for each loop iteration.

Figure 2 shows a typical MOLA procedure written for ReDSeeDS project. This procedure is a part of the transformation which builds sequence diagrams from requirements written using constrained language scenarios. The task for this procedure is to build a UML package and traceability link **for each** requirements package which contains appropriate requirements. Next, **every** requirement in **these** packages should be processed. Note the similarity of bolded linguistic constructs used in the description and MOLA constructs - the foreach loop and reference. This allows describing such algorithms very straightforwardly and easily in MOLA. The outer foreach loop (the bigger black frame) iterates through every requirement package, which has at least one requirement with a scenario. The loophead is the rule containing the loop variable (the uppermost rule in the loop). The loop variable (*pack: RequirementsPackage*) is used to explicitly denote the class to iterate through. A constraint which filters matched instances is given using the additional class elements (*existsUC* and *cls*). It restricts the iteration to requirement packages which contain a requirement represented by a constrained language scenario. This foreach loop contains also a loop body which consists of a rule and another foreach loop. They are executed in every iteration of the loop. The rule, which creates a UML package (*sPack* class element), places this package into the appropriate top-level package (*owningPackage* association link to *@p*), sets the name of the package and finally creates the traceability element (*ia* class element and *allocationSource* and *allocationTarget* association links). The nested loop is a typical approach in MOLA to iterate through contained elements. The inner loop iterates through all requirements (loop variable *uc*) which are in the matched requirements package (the reference *@pack*) and which contain a scenario (the class element *cls*).

2. Related Approaches of Pattern Matching

The closest "relative" to MOLA in the world of model transformation languages is Fujaba Story Diagrams from Fujaba Tool Suite [5,7]. Fujaba is a graphical model transformation language which uses imperative control structures and declarative patterns. The specification of patterns in Fujaba is almost identical to MOLA. There is a restriction on patterns in Fujaba - the pattern must contain at least one bound (previously matched) element. The graphical syntax, of course, differs for both languages, but that is obvious for independently developed languages. The most significant difference between the two is the foreach loop. Fujaba does not specify the loop variable and a loop is executed through all of the possible matches of the pattern. In MOLA only the distinct instances that correspond to the loop variable are iterated over. MOLA foreach loop is more readable and easier to use, because of the loop variable.

A different programming paradigm is used in the graph transformation language AGG [8], which is a typical example of a declarative transformation language. AGG does not have any imperative control structures, and rules that describe patterns are being executed independently. The only way to affect the execution order is to use layering. Each rule in AGG includes a pattern which is specified by LHS graph and NACs. NACs are used by declarative transformation languages mainly to distinguish already processed model elements. Negative patterns are used differently in MOLA because of the specific

loop construct. MOLA also has negative pattern elements, but they are used to express a "logical" negative condition. The graph transformation language PROGRES [2] is a textual graph transformation language where patterns (graph queries) are specified graphically. Patterns allow using similar and even richer options than previously noted transformation languages. The ordering of statements is managed by algebraic structures and PROGRES follows declarative PROLOG-like execution semantics. Graph transformation language VTCL (Viatra Textual Command Language), which is part of the VIATRA 2 framework [3], defines patterns using textual syntax. VIATRA offers broad possibilities for the pattern definition: negative patterns may be at arbitrary deep level; the call of a pattern from another pattern, and even recursive patterns are allowed; the language may work both with model and metamodel. The execution order of rules is managed by ASM (Abstract State Machine) language constructs which are purely imperative. VIATRA has a rudimentary graphical syntax of patterns, however it seems that whole expressiveness of the language may not be available there. Another textual graph transformation language, which has appeared in recent years, is GrGen [4]. The expressiveness of patterns in this transformation language is close to VIATRA. Transformation rules are combined using similar algebraic constructs to PROGRES (except the PROLOG-like execution semantics).

Almost all model and graph transformation languages that use similar pattern concepts as MOLA are forced to deal with pattern matching task. There are four most popular algorithms that are used by transformation language implementations to solve the pattern matching problem:

1. **Local search plan generation.** The search of the instances corresponding to pattern is started from a single instance, which is a potential match - it corresponds to some pattern element. We may say that the pattern matching has been started from the corresponding pattern element. Next, the adjacent instances are examined according to the given pattern. If the examined instances do not match or another valid match is needed, backtracking is required. A search plan is the order in which the potential matches are examined. In fact, a search plan is the sequence in which pattern elements are traversed in order to find a valid match. The search plan generation algorithm must examine various search plans and evaluate them in order to choose one that is least expensive. The algorithm uses a cost model of basic search operations. Then the search graph based on the pattern is built and weights are attached according to the cost model. At last, the appropriate ordering is determined from the graph.

2. **Reducing to CSP** (Constraint Satisfaction Problem[9]). The pattern matching problem is reduced to an equivalent CSP. Pattern elements are mapped to the variables and constraints. This enables to use all the techniques known in the area of CSP. The main techniques are related to the appropriate ordering of variables and efficient use of backtracking. Thus, in general both methods, local search plan generation and CSP, are quite similar, but CSP puts more emphasis on intelligent backtracking.

3. **Using relational database.** Using relational database reduces the pattern matching problem to taking advantage of the power of query optimization in relational databases management systems. The task is to choose an appropriate database schema to store the model and to generate the SQL query which returns the valid matches.

4. **Incremental pattern matching.** The core idea of incremental pattern matching is to make the occurrences of a pattern available at any time. It requires caching all occurrences of a pattern and incremental updating whenever changes are made. If this

requirement is met then the pattern matching is made in almost constant time (linear to the size of result set itself). However, the drawbacks are memory consumption and overhead on update operations.

PROGRES was the first transformation language addressing the pattern matching problem[10]. It uses the local search plan generation. PROGRES builds a pattern graph, where a node is built for each pattern element. Next, the operation graph is built, adding information about all operations that may be used by pattern matching. The cost of each search operation is derived from heuristic assumptions and knowledge on multiplicities of pattern elements. The best-first method is used to determine the search plan from the operation graph.

VIATRA has been implementing most of pattern matching algorithms. The relational database algorithm for VIATRA uses a separate table for each class [11]. An appropriate SQL query is used for finding the pattern matches.

The generation of local search plans also has been used by VIATRA[12]. The search graph is built for a pattern. An additional starting node is added to the graph. Directed edges connect the starting node to every other search graph node. Each edge of the pattern is mapped to a pair of edges in the search graph, expressing the bidirectional navigability. The cost model is obtained by analyzing the existing models of the domain, e.g. typical UML class diagram, if the UML class diagram is being transformed. The collected statistics illustrate an average branching factor of a possible search space tree, built when pattern matching engine selects the given pattern edge for navigation. Costs are added to the search graph and the minimum spanning tree (MST) is found with the starting node taken as the root node. The search plan is determined from MST.

An incremental pattern matcher[13] (RETE network) is constructed based on the pattern definitions. Before transformation the underlying model is loaded into incremental pattern matcher as the initial set of matches. The pattern matching is performed efficiently; however the changes should be propagated within the RETE network to refresh the set of matches.

The authors of VIATRA have introduced a hybrid pattern matching approach[14], which is able to combine local search and incremental techniques on a per-pattern basis. Two scenarios are proposed: design-time selection of the strategy by developer and run-time optimization based on monitoring of statistics (available memory or model space statistics).

GrGen uses a very similar local search plan generation method[15] to VIATRA. The plan graph (search graph by VIATRA) is built in a similar way, but the lookup operation for pattern edges is added. The cost model is built based on statistics collected from the host graph (model) just before the execution of the transformation. The costs are added, the MST calculated, and a search plan is determined in a way similar to VIATRA.

Fujaba uses a less advanced local search plan strategy[16]. The pattern matching in Fujaba is started from the bounded element (the requirement in Fujaba is to have at least one per pattern). If there is more than one bounded element, one is chosen arbitrary. Links to follow are chosen by priorities using the first-fail principle. Regardless of simplicity of this algorithm, the benchmark tests[17] show that this strategy works almost as good as more advanced algorithms.

AGG uses an algorithm, equivalent to the current pattern CSP[18]. This approach introduces variables for each pattern node and queries for each pattern edge forming the constraint graph. This graph is quite similar to the search graph in local search graph

generation technique. Variable ordering used in the area of CSP is essentially the same as the concept of the search planning.

The previous version of **MOLA** Tool was based on a fixed relational database schema as a model repository. A single SQL-query was built and executed for every pattern by MOLA interpreter [19]. However, the performance of a query optimization was not sufficient for large queries generated by the interpreter.

The latest implementation of MOLA uses a lower level model transformation language L3 [20] as the target language for local search plan generation. L3 provides the basic operations on models and its implementation is oriented to the maximum execution efficiency and performance. It is a purely imperative language where patterns are also expressed in an imperative way. At the same time it is powerful enough to serve as a target language for MOLA compiler. The details of pattern matching implementation of MOLA compiler is discussed in Section 3.

3. Domain-Specific Pattern Matching

Although pattern matching strategies described in the previous section are rather different, they all are based on the same principle - decreasing the number of instances needed to be checked. This principle is behind the optimization of SQL queries, the CSP variable ordering and, of course, behind the most popular pattern matching strategy - the local search plan generation. Using these methods the efficiency of pattern matching is dependent in a great degree on knowledge of actual number of instances in a model being transformed. Precise number of instances in a model or let's say *actual cardinalities* can be obtained by analysis of run-time models. It can be done just before the pattern matching like it is done in the case of GrGen [15]. However, this causes additional complexity for implementation and requires an interpreter-based solution for pattern matching implementation. Whole analysis and optimization should be done in runtime. An appropriate support of repository is needed too. Another way to get the actual cardinalities is the design-time analysis of typical run-time models. This method is described in [12]. The design-time analysis requires a lot of models to be processed in order to obtain a useful results. However, the availability of appropriate models is under question in a typical case.

Metamodel and pattern definition are also sources of information about the number of instances in models to be transformed. Of course, they do not show (and cannot show) the actual cardinalities. EMOF-like metamodeling languages allow specifying cardinalities for association ends. Usually one can specify that the navigation result may contain at most one instance (cardinalities 0..1 and 1) or many (cardinalities * and 1..*) instances. However, the possible number of instances of classes or more precise number of instances reachable from instance of given class by the specific association are missing in metamodels. Nevertheless, sufficiently efficient solutions which use just information from metamodels and different heuristics have been built, for example, the PROGRES solution [10]. The existing pattern matching implementations use just a general assumptions about metamodels and models being transformed. The complexity of pattern matching implementation would be much lower if the domain-specific information could be taken into account.

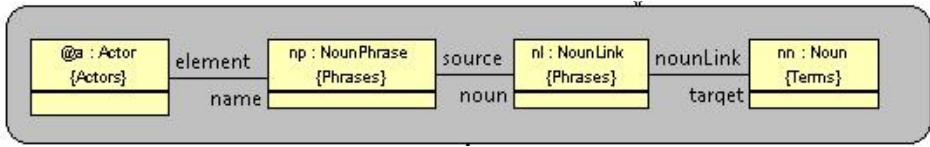


Figure 3. Pattern example - collecting nearby instances

3.1. MOLA Language Design Patterns for MDSD

As it has been mentioned in the previous chapters MOLA language has been designed to solve tasks specific to model-driven software development. How can this knowledge help to do an efficient pattern matching? What does it actually mean - a task specific to MDSD?

In MDSD models describe various aspects of a system under development. These models are well structured. Model elements form a tree structure according to the composition hierarchy defined by the metamodel. Typically all model elements are parts of the composition *tree* and can be reached by traversing it. For example, both languages (RSL and UML) used in the ReDSeeDS project describe typical MDSD models. In RSL constrained language scenarios are parts of requirements which in turn are parts of requirement packages. Similarly UML elements always are parts of some container element - attributes are parts of classes, classes are contained in packages, and so on.

MDSD tasks can be described as *compilation* tasks. Every (or almost every) element of the source model is traversed and an appropriate target element is created. Since the goal of MDSD process is to obtain a working system (it's a goal of every software development process) then this analogy is even more obvious. In fact, a requirement model is compiled to architecture and further to detailed design models in ReDSeeDS.

Another important property of MDSD tasks is a *locality* of transformations and conditions. It means that typically connected elements in a source model are transformed to connected elements in the target model. Therefore, even if models are described by hundreds of classes, a transformation of a single model element requires just the local neighbourhood of the element to be seen.

Keeping in mind the just described MDSD properties (tree-like models, compilation-like tasks, local transformations) we will study the typical MOLA patterns used in the ReDSeeDS project. To approximately estimate the volume of transformations written during the ReDSeeDS project we are giving some statistics. The model-based methodologies used in the project cover quite a large subset of UML being generated - UML class, activity, component and sequence diagrams. Both methodologies include several transformation steps. The first step for both methodologies is the transformation of requirements. The next steps are generation new UML models adding more specific details. There are ~350 MOLA procedures developed during the ReDSeeDS project. They include ~200 loops and ~800 rules that gives ~1000 pattern specifications.

A typical pattern used in the ReDSeeDS project is depicted in Figure 3. This pattern finds the name of an actor (names are coded as noun phrases in RSL). The example illustrates the locality of a transformation. The information needed for transformation (actor's name) can be reached by association links in a close distance. Note, that all associations leading from the Actor class to the Noun have cardinality "1" or "0..1" - each actor has exactly one name (represented by noun phrase), there is only one noun

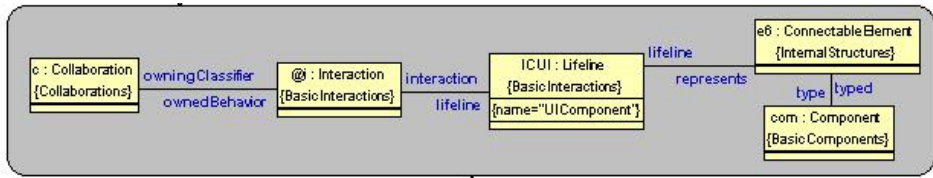


Figure 4. Pattern example - collecting nearby instances using additional constraints

link for each noun phrase and every noun link is connected to exactly one noun. Such structure (a reference and *low-cardinality* association links going away from it) is typical for patterns which collect data for the actual transformation.

Variation of the previous pattern is shown in Figure 4. This pattern describes the collecting of nearby elements of a UML interaction. The owning classifier and the component corresponding to the lifeline named "UIComponent" should be matched. Unlike in the previous example there is an association with cardinality "*" leading from the referenced element (to Lifeline). However, as we see in practice, typically there is only one model element in the model satisfying the given constraint and the "suspicious" association has a *low cardinality* in practice. In this case there are no more than 5-10 lifelines per interaction.

Figure 2 refers to the typical usage of loops in the ReDSeeDS project. Since RSL and UML model elements form a tree-based hierarchy, the transformation algorithms traverse model elements in the top-down style starting from the top elements of the hierarchy. The most natural way to describe such traversing is by using nested foreach loops referencing the previous loop variables. The pattern may contain additional class elements for collection of all necessary neighbourhood instances or specifying additional constraints on the existence of appropriate nearby instances. There are additional constraints in our example.

3.2. The Simple Pattern Matching Algorithm for MOLA

As we have got an insight what are the typical MOLA patterns for MDSD tasks, we can propose a simple (in the sense how complex is the implementation) algorithm, which is efficient for the patterns described above. The simple algorithm uses the following principles:

- if the pattern contains a reference class element, then the pattern matching starts from the reference (if there are more than one, then an arbitrary is chosen).
- otherwise the pattern matching starts from the loop variable in a loophead or from arbitrary chosen element in a normal rule.
- the pattern matching is continued with class elements accessible from already traversed class elements by association links.

Pattern matching in a regular rule is started from the reference class element, if such class element exists in the pattern. Though MOLA does not require the presence of a reference class element in the pattern, the practical usage of MOLA has shown that most of the regular rules contain it (see Figures 2, 3 and 4). Usage of imperative control structures causes the reuse of the previously matched instances, which are represented by the reference class elements in MOLA. This is one of the main reasons why such

simple optimization technique works almost as well as more sophisticated approaches. For example, the pattern depicted in Figure 3 is matched in constant time when the simple pattern matching strategy is applied. Another example was depicted in Figure 4. This pattern matches in linear time with regard to the number of lifelines in the given interaction, which is relatively *low*.

Use of reference class elements is natural also in loopheads. It is common to have a loop over, for example, all properties of a given class. This task can be easily described, using a single MOLA loop, where the pattern in the loophead is given using the reference class element and the loop variable. See the loophead of the inner loop in Figure 2 as a typical case. In this case the pattern matching is started from the reference element (*@pack*) reducing the search space dramatically. Of course, the path from the reference class element to the loop variable may be longer. The only restriction is that cardinalities of associations along the path (except one directly before the loop variable) should be "1" or "0..1". In a foreach loop without a reference in the loophead, the pattern matching is started from the loop variable in the loophead. The practical usage of MOLA has shown that typical tasks are naturally programmed using patterns, where actual cardinalities of association links leading from the loop variable are *low*. This causes the execution of the loop to work in almost linear time depending on the number of the instances corresponding to the loop variable.

Note the loophead of the outer loop in Figure 2. Though cardinalities of association links leading from the loop variable are "0..*", the pattern matching started from the loop variable is still efficient. Since class elements other than the loop variable provide the "existence semantics" (find first valid match), in practice this loop works also in linear time because almost all requirements are described using scenarios. In fact, this additional constraint is used to filter out those few cases where requirements are described using different means.

Note that this strategy does not even require the analysis of the cardinalities of meta-model elements at the same time remaining efficient in the practical usage. A similar pattern matching strategy is used also by Fujaba. The bound variable (reference class element in terms of MOLA), is even required by the pattern in Fujaba. However, the benchmark tests [17] have shown that this strategy performs as well as more sophisticated strategies. The same tests also have shown that an appropriate usage of the language constructs (transformation *design-patterns* used to improve Fujaba transformation) causes a significant positive impact on the performance.

We have tested the transformations on several sufficiently large software cases developed within the ReDSeeDS project. The total time of execution turns out to be almost linear with regard to the total number of constrained language sentences in the requirement scenarios specified in the RSL for the case. The patterns described above are the most typical patterns used in MOLA transformations for the ReDSeeDS project. The total amount of such patterns is about 95% of all patterns. Some specific sub-tasks require non-typical patterns which may cause insufficient pattern matching performance, however in practice they are performed on elements which are relatively low in number compared to the number of constrained language sentences. Thus, they do not affect the overall performance of pattern matching.

Domain-specific knowledge helped us a lot to reduce the complexity of pattern matching algorithm implementation. In fact, the algorithm has just three simple rules.

The key aspect that allows such simple algorithm is the identification of typical patterns used in MDSD-related tasks.

3.3. Refinement of Cardinalities

As we have seen the major role in the implementation of pattern matching is played by the actual cardinalities of model elements. Efficient pattern matching using the simple algorithm is possible mainly because the cardinalities are low for the key associations. However existing metamodeling languages cannot distinguish between low and high cardinalities. One reason is a lack of appropriate means in languages. Another reason is that there are no precise definitions of what *low* and what *high* cardinality is. In this section two main cardinality classes are introduced.

A metamodel element (class or association end) has a *low* cardinality if the number of corresponding instances is not dependent of total number of instances in the model. We may say that such cardinality is constant against the total number of instances in the model. For example, we can expect that in a UML class diagram a typical class will have about 5-10 properties, and this number is independent of the model size.

A metamodel element (class or association end) has a *high* cardinality if the number of corresponding instances is dependent of total number of instances in a model. For example, in a UML class diagram the number of typed elements for every type grows as the size of the class diagram increases.

Now taking into account the defined cardinality classes we can introduce a new means to MOLA metamodeling language and MOLA itself which allow to define cardinalities more precisely. We allow annotating classes and association ends in the metamodel and class elements and association link ends in patterns. The following annotations can be used:

- **SINGLE** - denotes that the class (or navigation result) has at most one instance. In fact, this annotation is used for classes, because cardinality "1" or "0..1" can be already specified for an association end.
- **FEW** - denotes that the class (or navigation result) has *low* cardinality.
- **MANY** - denotes that the class (or navigation result) has *high* cardinality.

Annotations made in the metamodel affect the pattern matching algorithm in every rule where pattern elements of the corresponding type are used. Annotations made in the pattern affect the pattern matching algorithm only in the scope of the corresponding rule.

What effect annotations do on pattern matching? The "SINGLE" annotated elements and links as well as references are preferred for the pattern matching. The "FEW" annotated elements and links are preferred over links that are not annotated. Links and elements that are not annotated will be preferred over links and elements with the "MANY" annotation. In fact, an annotation sets the priority on the pattern element. The lower the predicted number of instances is for the pattern element, the higher priority it gets for the pattern matching. The developer annotates metamodel elements during the development process of the metamodel. Since metamodeling requires the knowledge of the modelled domain, typically there are no problems to resolve actual cardinalities. It should be noted that annotations are optional - they are additional means to improve the efficiency of transformations.

Let us illustrate the usage of annotations. Figure 5 shows a pattern in a loophead where annotations help to find the best search plan. This loop iterates through every prop-

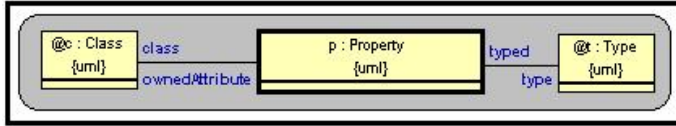


Figure 5. Pattern example - annotation use case

erty (p) of the given class ($@c$) having the given type ($@t$). The problem is that associations *ownedAttribute* and *typed* both have cardinality "*" and without additional information both are treated equally (un)efficient for pattern matching. However, in practice the average number of owned attributes for a class is by magnitude less than typed properties for a type. In fact, cardinality of attributes for a class is *low*, but cardinality of typed elements for a class is *high*. Therefore, adding annotations "FEW" and "MANY" to *ownedAttribute* and *typed* association ends accordingly solves the problem - navigation using the *ownedAttribute* association end is preferred over navigation using the *typed* association end. The annotation mechanism allows refining the existing means for describing cardinalities. It brings the domain-specific knowledge which is crucial for pattern matching implementation into metamodeling and model transformation languages.

4. Conclusions

We have made a review of pattern matching mechanisms for the most popular model transformation languages in this paper. There are several pattern matching approaches, but the most popular is the local search planning. In fact, it is the most universal strategy - it gives efficient results for different types of patterns. However, implementations of more advanced approaches are rather complex, although simpler strategies (like in case of Fujaba) frequently give similar results. Of course, that holds not for every use case, but mostly for **the domain** the transformation language is designed for. For example, MOLA is efficient for MDSD-related tasks, as the analysis of typical MOLA patterns in the ReDSeedS project has shown.

A great role for efficient pattern matching is played also by the constructs of the pattern used in the language. MOLA offers very natural means for describing MDSD-related tasks, the foreach loops combined with explicit reference mechanism. At the same time even the simple pattern matching algorithm which has been implemented for MOLA works efficiently in these cases. Thus, for the *compiler*-like tasks, where every element of a structured model (like UML) should be processed, MOLA can be used in a natural way with a high efficiency with very simple implementation of pattern matching.

The key aspect to success of the simple algorithm is finding out the actual cardinalities which are specific to the MDSD domain. The development of the simple pattern matching algorithm has revealed that metamodeling languages do not offer sufficient means to denote actual cardinalities. Therefore two new classes of cardinalities and the metamodel annotation mechanism which allows specifying the refined cardinalities in metamodel have been introduced. How the new annotation mechanism can be used to improve the efficiency of pattern matching algorithm has been also shown.

The future work is to identify model transformation domains - the areas where typical patterns are used. The most appropriate pattern matching approaches should be addressed for each domain. That would make the choice of an appropriate model trans-

formation language easier for a concrete task. Another possible research direction is the development of a domain specific annotation language which uses also other information, not only cardinalities. In fact, it means extending the metamodelling language with special features which capture information crucial for pattern matching.

References

- [1] Kalnins, A., Barzdins, J., Celms, E.: Model Transformation Language MOLA. In Aßmann, U., Aksit, M., Rensink, A., eds.: Model Driven Architecture, European MDA Workshops: Foundations and Applications, MDFA 2003 and MDFA 2004, Twente, The Netherlands, June 26-27, 2003 and Linköping, Sweden, June 10-11, 2004, Revised Selected Papers. Volume 3599 of LNCS., Springer (2004) 62–76
- [2] Schürr, A., Winter, A.J., Zündorf, A.: In: The PROGRES approach: language and environment. Volume 2. World Scientific Publishing Co. (1999) 487–550
- [3] Csertan, G., Huszerl, G., Majzik, I., Pap, Z., Pataricza, A., Varro, D.: VIATRA - visual automated transformations for formal verification and validation of UML models. In: Proceedings of 17th IEEE International Conference on Automated Software Engineering, IEEE Comput. Soc (2002) 267–270
- [4] Geiß, R., Batz, G.V., Grund, D., Hack, S., Szalkowski, A.: Grgen: A fast SPO-based graph rewriting tool. In: Proceedings of ICGT. Volume 4178 of LNCS., Springer (2006) 383–397
- [5] Fischer, T., Niere, J., Torunski, L., Zündorf, A.: Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java. In: Proceedings of TAGT. Volume 1764 of LNCS., Springer (1998) 296–309
- [6] Śmialek, M., Bojarski, J., Nowakowski, W., Ambroziewicz, A., Straszak, T.: Complementary use case scenario representations based on domain vocabularies. In: Proceedings of MoDELS. Volume 4735 of LNCS., Berlin, Heidelberg, Springer (2007) 544–558
- [7] University of Paderborn: Fujaba Tool Suite, <http://www.fujaba.de> (2010)
- [8] Taentzer, G.: AGG: A Tool Environment for Algebraic Graph Transformation. In Nagl, M., Schürr, A., Münch, M., eds.: Proceedings of AGTIVE. Volume 1779 of LNCS., Springer (1999) 481–488
- [9] Dechter, R., van Beek, P.: Local and global relational consistency. Theoretical Computer Science **173**(1) (1997) 283–308
- [10] Zündorf, A.: Graph Pattern Matching in PROGRES. In Cuny, J.E., Ehrig, H., Engels, G., Rozenberg, G., eds.: Proceedings of ICGT. Volume 1073 of LNCS., Springer (1994) 454–468
- [11] Varró, G., Friedl, K., Varró, D.: Implementing a Graph Transformation Engine in Relational Databases. Software & Systems Modeling **5**(3) (2006) 313–341
- [12] Varro, G., Friedl, K., Varro, D.: Adaptive Graph Pattern Matching for Model Transformations using Model-sensitive Search Plans. Electronic Notes in Theoretical Computer Science **152** (2006) 191–205
- [13] Bergmann, G., Ökrös, A., Ráth, I., Varró, D., Varró, G.: Incremental pattern matching in the viatra model transformation system. In: Proceedings of GraMoT, ACM (2008) 25–32
- [14] Bergmann, G., Horváth, A., Ráth, I., Varró, D.: Efficient Model Transformations by Combining Pattern Matching Strategies. In Paige, R.F., ed.: Proceedings of ICMT. Volume 5563 of LNCS., Springer (2009) 20–34
- [15] Batz, G.V., Kroll, M., Geiß, R.: A First Experimental Evaluation of Search Plan Driven Graph Pattern Matching. In Schürr, A., Nagl, M., Zündorf, A., eds.: Proceedings of AGTIVE. Volume 5088 of LNCS., Springer (2008) 471–486
- [16] Fischer, T., Niere, J., Torunski, L.: Konzeption und Realisierung einer integrierten Entwicklungsumgebung für UML. Master thesis, University of Paderborn (1998)
- [17] Geiß, R., Kroll, M.: On Improvements of the Varro Benchmark for Graph Transformation Tools (2007)
- [18] Rudolf, M.: Utilizing Constraint Satisfaction Techniques for Efficient Graph Pattern Matching. In Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G., eds.: Proceedings of TAGT. Volume 1764 of LNCS., Springer (1998) 238–251
- [19] Kalnins, A., Celms, E., Sostaks, A.: Simple and efficient implementation of pattern matching in MOLA tool. In Vasilecas, O., Eder, J., Caplinskis, A., eds.: Proceedings of DB&IS, Vilnius, IEEE (2006) 159–167
- [20] Barzdins, J., Kalnins, A., Rencis, E., Rikacovs, S.: Model Transformation Languages and Their Implementation by Bootstrapping Method. In Avron, A., Dershowitz, N., Rabinovich, A., eds.: Pillars of Computer Science. Volume 4800 of LNCS., Springer (2008) 130–145

A Kernel-level UNDO/REDO Mechanism for the Transformation-Driven Architecture

Sergejs KOZLOVICS¹*, Edgars RENCIS*, Sergejs RIKACOVŠ*, Karlis CERANS

University of Latvia, Faculty of Computing

Institute of Mathematics and Computer Science, University of Latvia

Abstract. The Transformation-Driven Architecture, TDA, is an approach for creating systems and tools. Can the UNDO/REDO functionality be easily (implicitly) introduced to the tools built upon TDA? The paper explains that this can be done at the level of TDA kernel in a universal way. We present the basic UNDO/REDO functionality as well as some its extensions.

Keywords: UNDO, REDO, UNDO metamodel, UNDO/REDO mechanism, TDA, Transformation-Driven Architecture

Introduction

In the ancient East it was considered that the king cannot err, thus his decrees could not be undone. That remained true even when later the king himself realized the unwanted consequences and wished the decree to be annulled. That was the case with king Darius from the book of Daniel. Without the UNDO functionality in software, users sometimes could feel like that king.

There is a robust solution to the UNDO problem, however. Evidently, the user can simply save the work at certain points. However, this naïve approach has at least two shortcomings. First, it leads to unnecessary wasting of resources, both time and space, since entire saving has to be done for all the states the user wants to be able to revert to. Second, the user is not able to UNDO changes in one part of the system, while preserving changes in another. Thus, a more intelligent solution is needed.

This paper presents the UNDO/REDO mechanism for the Transformation-Driven Architecture, TDA [1], which is a system- and tool-building approach (explained in Section 1). While our earlier explanation of the UNDO mechanism [2] has been based upon the current version of TDA, this paper focuses on the upcoming version of TDA, with its kernel playing an important role in the architecture. The proposed UNDO/REDO mechanism is a part of that kernel. The TDA UNDO mechanism is common for all tools intended to be used within the TDA Framework. When a tool is being developed, usually there is no need to think about UNDO. The UNDO and REDO capabilities are “miraculously” provided by the TDA Framework at runtime. And this is the main feature of the proposed UNDO mechanism (the basic ideas of the UNDO mechanism are explained in Section 2).

¹ Corresponding Author: Sergejs Kozlovics, Institute of Mathematics and Computer Science, Room 420, Raina blvd. 29, LV-1459, Riga, Latvia; E-mail: sergejs.kozlovics@lumii.lv.

* Partially supported by ESF project 2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004.

Of course, there may be cases when the built-in UNDO mechanism has to be adjusted or extended. For instance, the tool being developed may require several UNDO history streams (e.g., each diagram may have its own UNDO) with possible dependencies between those streams (UNDO in one diagram may require UNDO in another). Or, the TDA Framework may be extended, and new undoable actions (e.g., user actions, operations with external data, or changes to some exotic graphical presentation) have to be integrated within the TDA UNDO mechanism. Thus, we extend our UNDO mechanism for dealing with such issues (Section 3). First, we introduce multiple history streams and dependencies between them (Section 3.1). Interestingly, that many of the dependencies can be reasoned automatically, freeing the developer from that work. Next, we provide the API for new undoable actions and for specifying undoable states (Section 3.2). Besides, we show how the TDA UNDO mechanism can be extended to support multiple REDO branches (Section 3.3).

Although the proposed UNDO mechanism is intended to be used within TDA, the ideas presented in this paper can also be used in other model- and model-transformation-based tools.

1. The Transformation-Driven Architecture

The essence of the Transformation-Driven Architecture is depicted in Figure 1. In the center are model transformations that work with a model, which conforms to the system metamodel (the cloud). The system metamodel is divided into several parts. Some of these parts are called *interface metamodels* (ellipses), which are used by model transformations to communicate with engines. Engines (rectangles) provide certain services for transformations. For example, one engine may bring graph diagram editing capabilities (the Graph Diagram Engine [3]), while another may allow transformations to specify dialog windows, which are then displayed to the user (the Dialog Engine [4]). While there is a set of predefined engines (including the two just mentioned), new engines can also be added to TDA, when needed.

Engines and transformations access the model repository by means of the TDA kernel (the sphere in Figure 1). Actually, the kernel can work with several repositories (R1 and R2 in Figure 1), but provides a means, which allows transformations to access different repositories as if they were parts of one big repository. This is done by introducing a special class called *ProxyReference*. This class is a part of the kernel metamodel, which is stored in a separate repository (R0 in Figure 1) where the kernel stores its internal data. An instance of the *ProxyReference* class is a reference to some entity (object, class, association, etc.) in some repository. Instead of passing the actual entity to transformations and engines, the kernel passes the corresponding *ProxyReference* object. These objects are created when needed and deleted when needed and are managed by the kernel.

The communication between model transformations and engines is being performed by so called event/command mechanism. On certain moments (for instance, when a user performs some action) an engine creates an event object and calls the corresponding transformation to process that event. The transformation, in its turn, can create a command, which gives an instruction for the corresponding engine (e.g., to redraw a diagram). Thus, there are two predefined singleton classes *Event* and *Command* in the kernel metamodel. Each possible event type (right mouse click, double-click, keystroke, etc.) is a subclass of the *Event* class in the respective interface metamodel.

Similarly, commands are subclasses of the *Command* class. Event and command objects may be linked to other objects to specify the context.

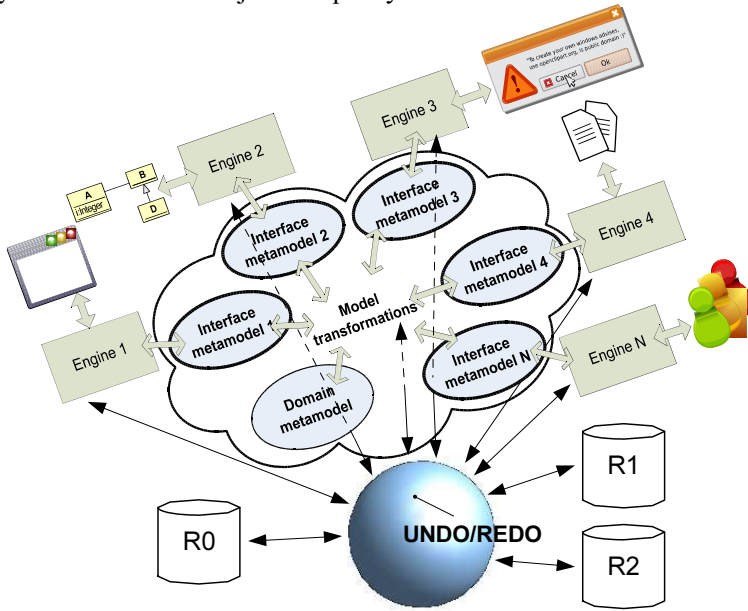


Figure 1. The Transformation-Driven Architecture.

The features of TDA mentioned above give us the following benefits regarding the UNDO/REDO functionality:

- Since access to the model data is by means of the kernel, we can use it as a proxy in order to hook model actions and to store them for later UNDO/REDO.
- Since usually several model actions are performed at once, they should also be undone/redone at once. The event/command mechanism helps to infer when a new “bundle” of actions starts, since it usually starts on certain events.

Thus, it is possible to implement the UNDO/REDO functionality directly in TDA, which relieves the transformation writers from thinking about UNDO/REDO. On the other hand, transformations can still adjust the default UNDO/REDO behaviour, when needed.

2. The Basics of the UNDO/REDO Mechanism

2.1. Basic Notions

When the user works with the system, the system changes its states. In TDA, the *system state* includes states of the repositories and states of engines and, possibly, other external states. When a “bundle” of logically bound actions is applied to the system, the system goes to a new state. For example, the user adds a new class to a class diagram. This invokes several actions, which include creating a box for representing the class (i.e., adding a box object to the repository) and attaching it to the diagram (i.e.,

creating a repository link from the box object to the diagram object). We do not consider intermediate states such as the state, when the class has been created, but has not been attached to the diagram yet.

The UNDO mechanism allows the user to revoke the last “bundle” of actions to be able to return the system to the state, which is equivalent to the previous (non-intermediate) state. We say “equivalent”, since the system state may be not the same as the original one. For instance, during UNDO/REDO deleted objects may be re-created, and their identifiers may change. The transformations and engines can still assume that the recreated objects are the same as the original ones, since transformations and engines do not address these objects directly, but by means of proxy references. The TDA kernel can ensure that proxy references used by transformations and engines originally are the same as after UNDO/REDO. However, proxy references may point to other (recreated) objects after UNDO/REDO.

In order to provide the UNDO/REDO feature, actions, which modify the state of the system, are trailed and then saved in the repository, where the kernel stores its internal data (R0 in Figure 1). We use the term *modifying actions* to denote the actions, which change the state of the system. These actions are “bundled” in *transactions*. Transactions represent differences between the system states. We call the memory area where the transactions are stored the *UNDO history*. To avoid the infinite recursion when trailing and storing changes in the UNDO history, the changes in the UNDO history itself are not trailed. Note, that we do not have to store in the UNDO history the complete snapshots of the system states along with transactions. Since transactions represent differences between the system states, executing inverse actions of a transaction in reverse corresponds to UNDO. Re-executing the transaction in the original direction corresponds to REDO.

We divide modifying actions into two groups:

- repository modifying actions, such as creating/deleting a class, an object, an association, a link, etc., in a model repository;
- external modifying actions: these are actions that modify states of the engines (outside the repository), actions that save information to external files, etc.

The first group of actions can be handled in a common, universal way by the TDA kernel. It simply acts as a proxy by hooking repository modifying actions and storing them in the UNDO history.

The second group of actions cannot be handled universally, since new engines with new modifying actions not known in advance may be added to TDA. However, it is possible to define a common API for these external modifying actions. Such an API would allow new actions to be easily integrated with the UNDO mechanism. We leave the explanation of this API until Section 3.2.

The TDA kernel itself cannot determine to which states UNDO must be able to revert the system. Thus, the kernel provides the `CreateUndoCheckPoint` function, which can be used by engines and transformations to mark the current state as a *checkpoint*, i.e., the state, to which the system will be able to revert. In reality, simply a new transaction is started, and from this moment new modifying actions are being attached to this new transaction. The `CreateUndoCheckPoint` function is usually called by engines on certain user events (e.g., events, which start some modification of a diagram). Thus, if the way the engines create checkpoints is satisfactory, the transformations do not need to set the checkpoints at all.

2.2. The Basic UNDO Metamodel

Having the basic notions explained, we introduce the basic metamodel for storing the UNDO history (Figure 2).

In its basic variant, the *UNDOHistory* consists of linearly ordered *Transactions*. The *currentTransaction* link specifies the transaction, to which modifying actions are being added. When neither UNDO, nor REDO has been performed yet, the current transaction is the one that was created when the last checkpoint was set. During UNDO and REDO the current transaction may change. For now we can assume that if some transactions have been undone, and the new transaction is started, the undone transactions (the “tail”) are deleted from the UNDO history.

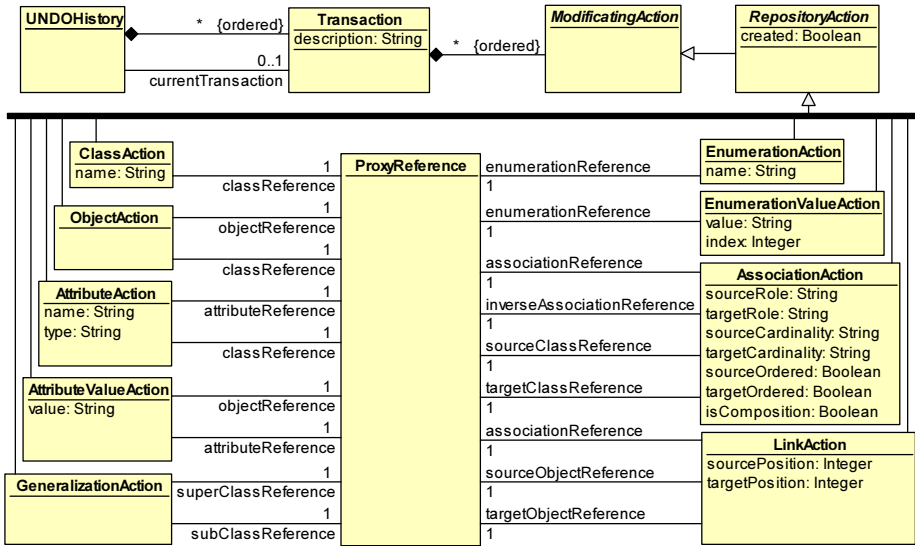


Figure 2. The basic metamodel for storing UNDO data.

Each transaction consists of *ModifyingActions*. For now, we consider only the repository modifying actions represented by the common superclass *RepositoryAction*. Each modifying action can either create or delete some element in the model, thus, we say that the delete action is an inverse of the create action, and vice versa. However, since each action may be undone and redone, we have to store information for both creation and deletion regardless of whether this is a creating or deleting action. Thus, in the UNDO metamodel, each repository action is grouped with its inverse in one class having a dual nature. And the *created* attribute in the *RepositoryAction* class specifies whether the original action created or deleted something.

When some object is being deleted, the kernel does not delete the corresponding instance of the *ProxyReference* class. This ensures that after UNDO, the proxy reference will be the same, although, it may point to another (recreated) object. Since the kernel passes to transformations and engines not real objects, but proxy references, re-creating objects during UNDO will not affect the transformations and engines — they can continue their work with the same proxy references.

However, when storing delete actions in the UNDO history, an important precaution has to be taken. If some object is deleted, its attributes, links, and probably other

objects (in case of composition) can also be deleted. The kernel has to store all these deletions in the UNDO history to be able to revoke all of them.

The *description* attribute of the *Transaction* class is used to describe the transaction. This description may be shown to the user as a hint before he is going to perform UNDO or REDO.

The metamodel in Figure 2 explains only internal structure for UNDO information not visible to the transformations and engines directly. Transformations and engines communicate with the TDA kernel through the API. Besides the already mentioned `CreateUndoCheckPoint` function this API contains the following functions:

- `SetUndoRedoDescription(description)` – specifies the description for the current transaction;
- `DisableUndo()` – stops writing UNDO history. This may be useful, for example, when some data is copied and stored in the repository to be pasted later. Switching off the UNDO mechanism for the time of copying allows the copied data remain in the repository regardless of UNDO/REDO;
- `EnableUndo()` – resumes writing UNDO history;
- `bool CanUndo()` – returns whether the current transaction can be undone;
- `Undo()` – undoes the current transaction;
- `bool CanRedo()` – returns whether the current transaction has the successive transaction, which can be redone;
- `Redo()` – makes the following transaction the current, and redoes it;
- `ClearUndoHistory()` – deletes all the transactions from the history.

3. Extending the UNDO/REDO Mechanism

In this section we extend the basic UNDO metamodel with the following features:

- Support for non-linear UNDO. Such a necessity may arise, for example, when working with several diagrams. The diagrams may be independent or not, and the non-linear UNDO can deal with dependencies.
- A way for storing external modifications and undoing/redoeing them.
- Support for REDO branching, i.e., preserving the REDO list, when modifications are made. Modifications then are stored in a new REDO branch.

3.1. Non-linear UNDO: Multiple UNDO History Streams and Their Dependencies

Assume the system state contains two independent diagrams A and B. The user of the system modifies Diagram A and then spends two hours modifying Diagram B. After that, he wants to undo the changes in Diagram A. If the UNDO history was linear (as in the basic UNDO metamodel), the user would have to undo his two-hour work before he could undo changes in Diagram A. Intuitively, we want to have separate UNDO histories for the diagrams A and B here. However, in certain cases dependencies between the diagrams may arise. For example, Diagram A may contain some element E, which is referenced in Diagram B. If E is deleted from Diagram A during UNDO, the changes, which created references to E in Diagram B should also be undone. In this section we describe a generic solution for sharing the common UNDO history between several presentations that may or may not depend on each other.

Let us look more narrowly at the system. It may contain different presentations (e.g., diagrams) that may be added or removed from the system at runtime. Each such presentation contributes some data to the system state. Thus, we can speak about presentation states, which are parts of the system state. When the system state changes during some transaction, this may affect also some presentation states, but leave other presentation states unchanged. That is, the presentation state may remain the same while transactions being performed do not affect it, until some transaction eventually changes the presentation state transforming it to a new state. We say that the sequence of different states for the given presentation forms the *UNDO history stream*. Each presentation state, or *stream state*, except the initial one, is associated with exactly one transaction, which led to it.

Figure 3 depicts how the notion of history stream is added to the basic UNDO metamodel. The *UNDOHistory* now may contain several *HistoryStreams*. Each history stream has a unique identifier (*id*). As was explained above, the history stream consists of *StreamStates*. Each stream state in the history stream (except the initial one) is associated with a transaction, which led to that stream state. Stream states are ordered according to the sequence of their appearance. One of the states is the current (role *currentState*).

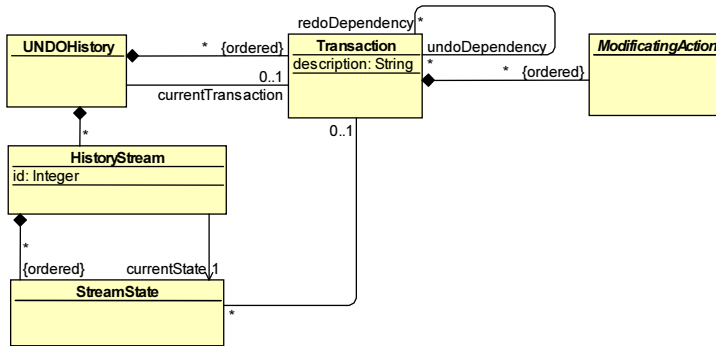


Figure 3. Extension of the UNDO metamodel introducing history streams.

UNDO now is invoked not at the level of the whole UNDO history, but at the level of one history stream (a “global” history stream, which trails all the transactions, can be introduced, when needed). Positions in the UNDO history for each history stream are now determined by the corresponding *currentState* links, and the order, in which UNDO/REDO is performed, is now determined by the composition between *HistoryStream* and *StreamState* classes. Transactions are simply collected in the UNDO history, and the *currentTransaction* link now simply points to the transaction, to which modifying actions are added. If the transaction changes some stream state, a new stream state is associated with that transaction.

To handle dependencies, we have to distinguish between the two their kinds, first. These are implicit and explicit dependencies.

Implicit dependencies are dependencies implied from the association between transactions and stream states. For instance, three UNDO streams from Figure 4 depend on each other, since UNDO in one history stream will cause undoing transaction T, and, thus, the other two history streams will change their states.

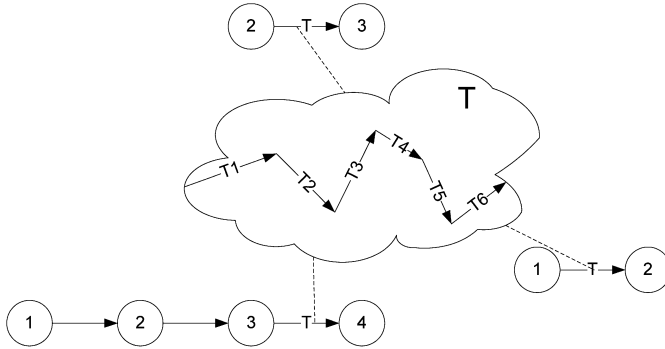


Figure 4. Transaction T (consisting of modifying actions T1-T6) is associated with several history streams (chains of circles). Circles correspond to stream states, and the arrows between them show the order. Transaction T led the topmost stream to State 3, the leftmost stream to State 4 and the rightmost stream to State 2.

Explicit dependencies are dependencies between different transactions. For that reason we introduce the *undoDependency/redoDependency* association. The semantics is as follows: when some transaction T is being undone, then all UNDO-dependent transactions (role *undoDependency*) are also undone (if not undone earlier). Similarly, when T is being redone, the corresponding REDO-dependent transactions (role *redoDependency*) are also redone. Explicit dependencies are always bi-directional: in one direction they are UNDO-dependencies, while in another direction they are REDO-dependencies.

Explicit dependencies may be added manually or automatically. Automatically added dependencies allow the developer not to think much (or at all) about specifying dependencies for correct UNDO behaviour. In this section we describe automatic explicit dependencies reasoned from repository actions. The next section will introduce automatic explicit dependencies reasoned from external states.

When the TDA kernel is processing repository modifying actions, it can reason about certain relations between transactions, and add corresponding explicit dependencies. For example, when a link between two objects is added in the current transaction C, the TDA kernel can find transactions A and B, where those two objects were created. Then, evidently, there are two dependencies: between C and A, and between C and B. When, for example, A is being undone, C also has to be undone, since the link cannot exist without its end object. Similarly, when C is redone, A also has to be redone: A has to re-create the end object for the link, which will be re-created in C.

Table 1 lists, which dependencies can be reasoned automatically. We show only REDO-dependencies referring to earlier transactions, which have to be redone when the current transactions is redone (the corresponding UNDO-dependencies will be added automatically, since dependency links are bi-directional). If there are several transactions with the given criteria, we assume the REDO-dependent is the most recent transaction.

Table 1. Dependencies: how they can be reasoned automatically.

Repository action	REDO-dependent transactions (if can be found in history)
ClassAction (created)	When the class with the same <i>name</i> was deleted.
ClassAction (deleted)	When this class was created.
ObjectAction (created)	When the class with the given <i>classReference</i> was created.
ObjectAction (deleted)	When this object was created.
AttributeAction (created)	When the class with the given <i>classReference</i> has been created. When the attribute of the given class with the same <i>name</i> was deleted.

Repository action	REDO-dependent transactions (if can be found in history)
AttributeAction (deleted)	When this attribute was created.
AttributeValueAction (created)	When the attribute was created. When the object was created.
AttributeValueAction (deleted)	When this attribute value was set/created.
AssociationAction (created)	When the class corresponding to the <i>sourceClassReference</i> was created. When the class corresponding to the <i>targetClassReference</i> was created. When the association from the class corresponding to the <i>sourceClassReference</i> with the same <i>sourceRole</i> was deleted.
AssociationAction (deleted)	When this association was created.
LinkAction (created)	When the association with the given <i>associationReference</i> was created. When the object corresponding to the <i>sourceObjectReference</i> was created. When the object corresponding to the <i>targetObjectReference</i> was created. When the link corresponding to the <i>associationReference</i> was deleted between objects corresponding to <i>sourceObjectReference</i> and <i>targetObjectReference</i> .
LinkAction (deleted)	When this link was created.
EnumerationAction (created)	When the enumeration with the same <i>name</i> was deleted.
EnumerationAction (deleted)	When this enumeration was created.
EnumerationValueAction (created)	When the enumeration corresponding to the <i>enumerationReference</i> was created. When the given <i>value</i> of the enumeration has been deleted.
EnumerationValueAction (deleted)	When the given <i>value</i> of the enumeration has been created.
GeneralizationAction (created)	When the class corresponding to the <i>superClassReference</i> was created. When the class corresponding to the <i>subClassReference</i> was created. When the same generalization was deleted.
GeneralizationAction (deleted)	When this generalization was created.

When the kernel is processing some repository modifying action, it simply finds the corresponding REDO-dependent transaction (if any), and creates the corresponding dependency between that transaction and the current transaction. In order to be able to find out the dependent transaction, the TDA kernel may store a map, which maps descriptions of the actions in the UNDO history to the corresponding transactions.

To support history streams, we add the following functions to the TDA kernel API:

- `CreateUndoHistoryStream(id)` – creates a new history stream with the given *id*;
- `ClearUndoHistoryStream(id)` – deletes all the states in the history stream except the last one, which becomes the current state;
- `DeleteUndoHistoryStream(id)` – deletes the given history stream with all its states (should be called, when the corresponding presentation is removed from the system).

Besides, we add the history stream *id* parameter to the functions `CanUndo`, `Undo`, `CanRedo` and `Redo`.

3.2. Adding New Actions and States

Up to this point, we assumed there are only repository modifying actions. From now we suppose that engines and transformations may also make changes in other parts of

the system such as some external database or a graphical window with visualization that needs to be updated after UNDO or REDO. In this subsection we extend the UNDO mechanism with the ability to register external actions and states in the UNDO history.

When some engine creates a new presentation (e.g., a diagram) on the screen, it may also create an UNDO history stream for that presentation. When the presentation changes, it may inform the TDA kernel about that change in two ways:

- Inform that a state has been changed. This way is suitable for cases when many changes are performed at once, or when one change affects many aspects of the presentation encoded in one state (e.g., changes many coordinates).
- Inform that some undoable action has been performed. This is suitable, when the action performs some simple function similar to a repository action, or when this action affects only one (or just some) aspects of the presentation.

In order to support external states and actions we add several virtual methods to the classes *StreamState* and *ModifyingAction*.

In order to describe external states, an engine has to create a subclass of the class *StreamState* and implement the following virtual methods:

- *revoke()* – this method is called by the kernel, when the given state has to be removed from the presentation;
- *revert()* – this method is called by the kernel, when the presentation has to renew its state. Methods *revoke()* and *revert()* are called in pair – the first one for the old state and the second – for the new state. When the new presentation simply overrides the previous, only the *revert()* method has to be implemented;
- *save()* – this method is called, when the whole work is saved on disk. The given stream state may also be saved at this time. This allows the UNDO history to be renewed when the work is opened next time. If the *save()* method is not implemented, then not only this state, but, probably, other states (earlier states in the stream, or states from dependent transactions) and actions (e.g., actions from dependent transactions) are cut off from the UNDO history and not saved, too;
- *load()* – this method is called to load the saved state from disk;
- *destroy()* – this method is called, when the memory occupied by the state has to be freed (i.e., when the state is being removed from the UNDO history, or the user closes the work).

When a presentation (e.g., a diagram) requiring its own UNDO history stream is created, the engine may inform the TDA kernel about the initial external state of that stream by calling the function

```
CreateUndoHistoryStream(<history stream id>, <initialState>),
```

where *<initialState>* is the object of the corresponding subclass of *State*.

When the presentations changes its state, the engine calls

```
RegisterUndoStreamState(<history stream id>, <newState>),
```

and the TDA kernel automatically associates the new state with the current transaction. The *<newState>* becomes the current state in the given history stream.

The function

```
TruncateUndoHistoryStream(<history stream id>, <state>)
```

may be used to ask the kernel to cut off from the given history stream all the states before the `<state>`. This may be useful, when the engine sees that the memory limit for storing its states has reached its maximum value, thus, old states should be deleted to free memory for newer states.

Similarly to external states, we also introduce virtual methods for external actions. The methods `save()`, `load()` and `destroy()` in the class *ModifyingAction* are similar to the corresponding methods of the class *StreamState*. The difference is in methods `undo()` and `redo()`, which are used in the class *ModifyingAction*. Unlike the methods `revoke()` and `revert()`, which are called in pair when the state changes, for actions only one method – either `undo()` or `redo()` – is called depending on whether the actions is being undone or redone.

When an external action is being performed for the first (original) time, the engine has to register it in the UNDO history by calling the function

```
RegisterUndoableExternalAction(<action>),
```

where `<action>` is an object of the corresponding *ModifyingAction* subclass representing the external action. This action is added to the current transaction.

Reasoning Automatic Dependencies from External States. Assume a diagram, which initially was in State 1, has been brought by transaction T_1 to State 2, and then, by transaction T_2 , to State 3. Thus, State 1 contains changes neither of T_1 , nor of T_2 , State 2 contains changes of T_1 and not of T_2 , while State 3 contains changes both of T_1 and T_2 . Then it is impossible to undo T_1 independently of T_2 since there is no state with changes of T_2 without changes of T_1 in the UNDO history.

This observation allows the TDA kernel to create another kind of automatic explicit dependencies: when the kernel is notified about changes from States 1 to State 2 and from State 2 to State 3, it automatically creates a dependency between the transactions T_1 and T_2 .

When not external states, but external actions are registered, there remains a possibility to undo transaction T_1 independently on T_2 if no other dependency between them exists.

3.3. Adding Support for Multiple REDO Branches

An important property of the UNDO mechanism is the ability to revert to any recent state. However, in classical UNDO implementation when the history stream is a list, a modification after UNDO causes the “tail” of the history to be cleared, thus, the user may lose the ability to revert to certain states. Instead of clearing the “tail”, we can create a new history branch for new modifications. If the user reverts to the same state several times, and starts new modifications, then several branches may arise.

One of these branches is the current (later, the current branch status may go to another branch, as will be explained below). The modifications from this branch will be applied, if the user clicks the REDO button. We call the state, to which the current branch leads, the *next state*. The branches before the current branch lead to the *early next states*, and the branches after the current branch lead to the *late next states*. According to these terms, we replace the composition between the *HistoryStream* and *StreamState* classes in the UNDO metamodel by three forward unidirectional associa-

tions *earlyNext*, *next* and *lateNext*. We add also one backward unidirectional association *previous* (Figure 5).

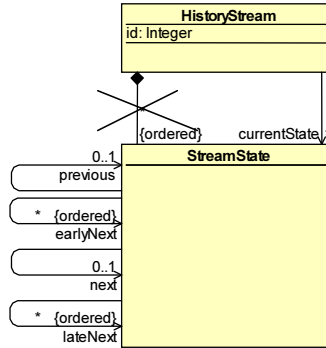


Figure 5. Replacing the composition between *HistoryStream* and *StreamState* with unidirectional associations to support REDO branching.

In Figure 6, if the history stream is in State 2 (state numbers correspond to time moments when the state has been initially reached before UNDO or REDO), and if the current next state is State 8, then the early next states are State 3 and State 5, while State 10 is the late next state.

One may think that in order to support branching, we have to add additional buttons to the standard two “Undo” and “Redo” buttons. Yes, we can do that. However, we can still continue using only two buttons, but they will change their behaviour slightly. We call them “Smart Undo” and “Smart Redo” buttons.

If there are early REDO branches, the “Smart Undo” button goes to the most recent state among the states in these early branches (see Figure 6(a)). If there are no early REDO branches, the “Smart Undo” goes back like the ordinary “Undo” button.

If there is the current branch, the “Smart Redo” button simply goes forward like the ordinary “Redo” button. If there are no branches, the “Smart Redo” button goes to the state before the next nearest branch (see Figure 6(b)).

This behaviour of the “Smart Undo” and the “Smart Redo” buttons ensures that the user can traverse all the states. Besides, the same state may be visited several times depending on the number of branches outgoing from that state.

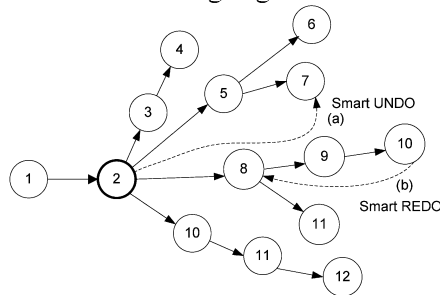


Figure 6. REDO branching (transactions are not shown) and behaviour of “Smart Undo” and “Smart Redo” buttons.

4. Related Work

Microsoft DSL Tools [5] uses the domain model framework, which supports UNDO and REDO. Although UNDO/REDO logs the changes of the model (similar to our repository modifying actions), it supports also propagation of changes, which allows the corresponding presentations to be updated according to the current model state. However, when describing rules for propagation of changes, the developer has to take some care depending on whether the changes are propagated during UNDO/REDO or not: repository changes must not be propagated during UNDO/REDO, while external changes still have to be propagated. In our approach, external changes may be added to the UNDO history as external actions, thus the TDA kernel will call the corresponding *undo()* function to undo external changes, while no such function will be called for repository changes during UNDO.

A good example of a thought-out solution to the UNDO problem is Eclipse [6], on which a graphical editor building platform GMF [7] is based. A single UNDO history can contain operations, and a context may be associated with each operation. This is similar how, in our terms, history streams (~contexts) are associated with transactions (~operations). In Eclipse, an operation approver may be assigned to a operation, which is consulted before undoing/redoining that operation. An approver, for example, may forbid undoing an earlier operation while later operations have not been undone. In our approach, undoing within the same history stream is linear, while, at the level of the whole UNDO history, earlier transactions may be undone without undoing later transactions. One may choose between creating the single history stream for totally linear UNDO and creating an individual history stream for each transaction. We also allow creating dependencies between transactions, which force several transactions to be undone/redone at once. Certain dependencies may be reasoned automatically, so the developer does not have to specify them manually.

An interesting implementation of UNDO in Collaborative Modeling Tool is described by D. English [8]. Two special commands are introduced there: *Save* and *Cancel*. The *Cancel* command returns the system to the state when the last *Save* was performed. Interestingly, that *Cancel* itself is added into the UNDO stack, and, as a consequence, the *Cancel* action by itself can be undone. This allows “forks” of user actions, starting from the last saved state, to be created. This resembles our REDO branching described in Section 3.3. And the process of undoing the *Cancel* command resembles the behaviour of our “Smart Undo” button.

There is also a tool ATOM³ [9], where UNDO is implemented by simply saving the model. This seems to be a good solution specifically for that tool since UNDO has not been built-in there originally.

An interesting application of UNDO for debugging purposes is described in the paper of Hartmann and Sadilek [10]. Assume some model (e.g., Petri net) is being executed and at some step an incorrect behaviour is found. When the execution semantics has been corrected, UNDO can be used to revert the model to the last correct state, and the execution may be re-started from that state instead of performing all the execution from the beginning.

The concept of worlds [11] is a way of controlling the scope of side-effects. The states of the system form a tree, where each node (except the initial one) handles access to its parent’s data and stores read and modified values. UNDO can be performed by switching from a child-world to its parent world. Since worlds form a tree, a REDO-branching is also supported in this concept.

An interesting application of constraint evaluation in order to determine UNDO dependencies has been proposed recently [12]. This approach allows inferring the dependencies at the time of UNDO depending on the model elements the user wants to revert to previous states. In our approach dependencies have to be set before UNDO, although they may be calculated by evaluating some kind of constraints.

5. Conclusion

Certain ideas presented in this paper have been implemented (with certain distinctions) in the TDA-based tool-building platform GrTP [13]. An interesting fact is that incorporating UNDO into TDA did not require modifying transformations. However, the Graph Diagram Engine was modified to create checkpoints on user events, to register its states within the UNDO history, and to provide “Undo” and “Redo” buttons. In current applications UNDO works without observable delay.

An interesting question is: How to implement the UNDO/REDO functionality when multiple users edit the same model concurrently? Current results in that area [14] give us hope that such multi-user UNDO is possible. However, incorporating it into the UNDO mechanism proposed in this paper requires additional study.

References

- [1] J. Barzdins, S. Kozlovics, E. Rencis. The Transformation-Driven Architecture. Proceedings of DSM'08 Workshop of OOPSLA 2008, Nashville, Tennessee, USA, 2008.
- [2] S. Kozlovics, E. Rencis, S. Rikacovs, K. Cerans. Universal UNDO Mechanism for the Transformation-Driven Architecture. Proceedings of Ninth Conference on Databases and Information Systems 2010, Riga, Latvia, 2010.
- [3] J. Barzdins, K. Cerans, S. Kozlovics, E. Rencis, A. Zarins. A Graph Diagram Engine for the Transformation-Driven Architecture. Proceedings of MDDAUI 2009 Workshop of IUI 2009, Sanibel Island, Florida, USA, 2009.
- [4] S. Kozlovics. A Dialog Engine Metamodel for the Transformation-Driven Architecture. To be published in Scientific Papers, University of Latvia, 2010.
- [5] S. Cook, G. Jones, S. Kent and A.C.Wills: Domain-Specific Development with Visual Studio DSL Tools, Addison-Wesley, 2007.
- [6] The Eclipse Project. <http://www.eclipse.org>
- [7] A. Shatalin, A. Tikhomirov. Graphical Modeling Framework Architecture Overview. Eclipse Modeling Symposium, 2006.
- [8] D. English. To UNDO or not to UNDO. <http://dougjon.blogspot.com/2009/01/to-undo-or-not-to-undo.html>
- [9] J. de Lara, H. Vangheluwe, M. Alfonseca. Meta-Modeling and Graph Grammars for Multi-Paradigm Modeling in ATOM3. Software and System Modeling, 3(3), pp. 194-209, 2004.
- [10] T. Hartmann, D. A. Sadilek. Undoing Operational Steps of Domain-Specific Modeling Languages. Proceedings of DSM'08 Workshop of OOPSLA 2008, Nashville, Tennessee, USA, 2008.
- [11] A. Warth, Y. Ohshima, T. Kaehler, A. Kay. Worlds: Controlling the Scope of Side Effects. Viewpoints Research Institute Technical Report TR-2010-001, 2010.
- [12] I. Groher, A. Egyed. Selective and Consistent Undoing of Model Changes. Proceedings of MODELS 2010, Oslo, Norway, 2010.
- [13] J. Barzdins, A. Zarins, K. Cerans, A. Kalnins, E. Rencis, L. Lace, R. Liepins, A. Sprogis. GrTP: Transformation Based Graphical Tool Building Platform. Proceedings of MDDAUI 2007 Workshop of MODELS 2007, Nashville, Tennessee, USA, 2007.
- [14] C. Sun. Undo As Concurrent Inverse In Group Editors. ACM Transactions on Computer-Human Interaction, 9(4), pp. 309-361, 2002.

On Views on Metamodels

Edgars RENCIS*

*Institute of Mathematics and Computer Science,
University of Latvia,
Riga, Latvia*

Abstract. Several metamodels are often introduced when looking at the same fragment of the real world from different points of view. This process involves multiplying data to be instances of those metamodels thus enforcing the redundancy and complicating the maintenance. In this paper, a mechanism for defining (read only) views on arbitrary metamodels is presented. Having the mechanism, a user is allowed to work with the view metamodel and its instances by the means of a model transformation language while at the same time only instances of the base metamodel are kept physically in the memory. Thus, the proposed view definition mechanism does not introduce a redundancy of data. Also, the basic ideas of compiling model transformation programs from the view metamodel to the base metamodel are outlined here.

Keywords. Views on metamodels, redundancy, model transformations, compilers, graphical tool building.

Introduction

The very basic principle of relational data bases and data storage is that of reducing the redundancy of data. Every piece of information must be stored in no more than one location in order to save memory space and, what is more important, ease the maintenance of data integrity. However, in different applications it is often necessary to look at the data from different points of view. That is why the mechanism of views has been implemented in relational data bases.

If changing the way of storing data from a relational data base to a metamodel-based repository, the main principles stay put. A well-established language for building metamodels is MOF (Meta-Object Facility [1]). It is a very powerful language, although its most exotic features are used rather rarely. So we will confine ourselves to EMOF (Essential MOF [1]) in this paper – the subset of MOF including all the most widely used features. So, when using a term “metamodel” in this paper, a model of EMOF is to be thought of. The common way of transforming instances of one metamodel to instances of another metamodel is being done by means of model transformation languages.

The principle of reducing the redundancy is considered to be self-evident when building metamodels for any particular problems. However, a need for looking at the data from different points of view arises in practical applications. A good examples of the case can be found in the field of graphical tool building (e.g., metacase tools like MetaEdit+ [2,3], Microsoft DSL Tools [4], Eclipse GMF [5], METAclipse [6], or GrTP

* Partially supported by ESF project 2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004

[7]). Here, it is often a routine to have several metamodels coding about the same information for different purposes – domain, presentation, editing, etc. If all these metamodels together with their instances (further called – models) are stored physically in the memory or elsewhere, some kind of synchronization needs to be performed, and that can be really hard in some cases. Besides, data redundancy occurs in the case.

Another example is that of different needs for different users. For example, having a concept of flowchart diagrams in our mind, we can try to put it in different metamodels. If we start thinking of such a metamodel, we would perhaps initially come out with some metamodel having a class “Element” with several subclasses for different types of elements – “Start”, “Stop”, “Statement”, “Condition”. Sequence of elements could be encoded in some “previous/next” link between instances of the class “Element”. Then, trying to refine the metamodel, we would manage to build something more detailed (e.g., forming two new subclasses of “Element” – “StartElement” and “EndElement” and changing the “previous/next” association accordingly). On the other hand, we could be satisfied with the “default” metamodel for everything (having only one class “Element” and one association “previous/next”) in some cases. It is clear that we can code every particular flowchart as an instance of any of those metamodels. Or – we can look at every flowchart through the glasses of any flowchart metamodel. To answer the question of choosing the right one for our specific needs involves knowing the context of the particular case.

A simplified ontology of a university can serve as another example of using views. One can need to define a derived class “Good Student” from the class “Student” of the ontology saying that a student belongs to the class “Good Student” iff he has collected certain amount of credit points and passed all the mandatory courses.

Usually, all the necessary information is present to us in a form of either one big metamodel or several smaller ones. However, not always we are able to understand to which parts of the metamodel must we look or which parts to interpret in some other manner in order to extract the needed parts of information from it. We can have some mental image of the particular domain in our minds, and we have to map it to the metamodel in our hands. That is not always a trivial task.

Against this background, two main problems arise, salvation of which is offered in the paper:

1. to come out with a method for defining such views on EMOF models without introducing data redundancy;
2. to find a way, how to allow a model transformation language to be able to operate with views.

This paper is organized as follows. A description of a concept of a view is given in Section 1. It consists of two parts, each described in a separate subsection. Since avoiding redundancy is one of the key reasons in making views, not all of the information is actually been kept in the memory. Therefore, some actions need to be done in order to allow model transformation languages to be able to work with this information. This is explained in details in Section 2. The demonstration of the approach on a concrete example is given in Section 3. Then, in Section 4, some use cases of the view mechanism in the field of graphical tool building are outlined. At the end, some related work is inspected in Section 5.

1. Notion of a View on Metamodel

In this section, a concept of a view on a metamodel is given. Also, a general mechanism of how to build views on an arbitrary metamodel is sketched here.

1.1. Relation ON

A metamodel of an oriented graph (see Figure 1, left part – “Base Metamodel”) will be used to explain the basic ideas. This will serve as a base metamodel on which a view will be built.

Let us suppose we want to build a view according to the following statements:

1. we want to look at all nodes in the view;
2. we want to look only at those edges that are visible (the value of the attribute “isVisible” is equal to *true*);
3. we do not want to have the class “EdgeEnd”, we want to use direct edge connections to nodes instead;
4. we want weight to be shown as an attribute of the edge instead of having it as a separate instance, hereto showing weight as single attribute (named – *label*) concatenating its value with its unit;
5. we want to introduce some renaming:
 - a. classes for nodes and edges would be called “ViewNode” and “ViewEdge”, respectively;
 - b. the caption of a node would be called “name” instead.

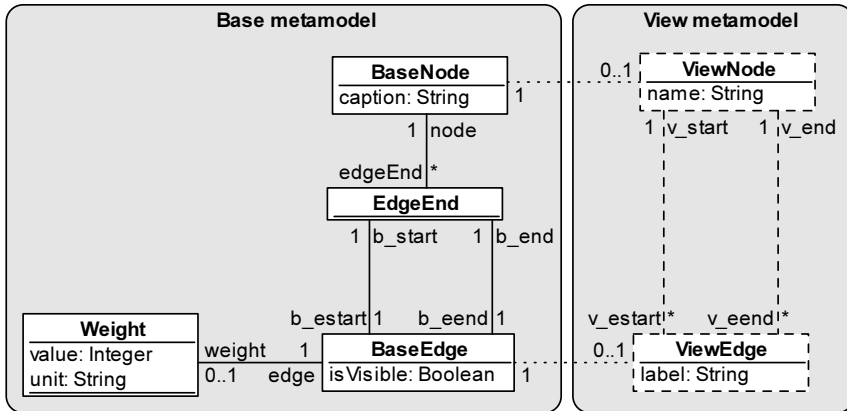


Figure 1. A metamodel of an oriented graph – the view metamodel connected to the base metamodel.

Now it is possible to construct a metamodel satisfying all the needs highlighted in the abovementioned statements (see dashed classes and associations in Figure 1 forming the view metamodel). View classes “ViewNode” and “ViewEdge” have been defined ON base classes “BaseNode” and “BaseEdge” (see dotted associations in Figure 1 connecting base metamodel with the view metamodel), and two new associations as well as two new attributes have been introduced.

It is very important to notice that for every class of the view metamodel, there exists exactly one class of the base metamodel to which it “belongs”. Therefore, an

expression “class A is defined **ON** class B” is used meaning we have a function **ON** from the set of view classes to the set of base classes. This property of a view will be used later. It must be clear that every graph that can be coded as an instance of the base metamodel can be as well coded as an instance of the view metamodel. Thus, we can look at every “base-coded” graph through a view in a form of the given view metamodel (consisting of only two classes). Having a particular instance of the base metamodel, we can transform it to an instance of the view metamodel, as long as we are familiar with the abovementioned statements. It must be noticed that this transformation does not necessarily works in the opposite direction (e.g., the attribute *label* might not be decomposed into *value* and *unit* without extra knowledge).

1.2. The Set of Graph Traversal Methods

Since avoiding redundancy was the key factor of making views, it must be clarified that neither the metamodel of the view nor the data themselves will actually be kept in the memory (that is why the view metamodel was dashed in Figure 1). Since only “real” instances (the ones of the base metamodel) exist there, the notion of the view (the view definition statements like the ones mentioned in Section 1.1) has to be attached somehow to this really existing base metamodel. Our goal here is to be able to traverse a “virtual” view model despite the fact there are no view instances in the repository. So, we would actually traverse some base model leaving an impression as if we were traversing the desired view model.

The basic idea here is to define a set of methods to be attached to each class of the base metamodel **ON** which a class of the view metamodel is defined. Calling these methods for concrete base objects will then allow one to get some information about the view classes/associations/attributes in the context of the object. For example, when traversing the instance graph of the metamodel presented in Figure 1 (left part) by means of some model transformation language, one can stop at some particular object p of, say, class “BaseNode” and ask: “Would I see p as an instance of “ViewNode” if I looked to my graph through the view metamodel?” In order to answer the question, we need to know the statements mentioned in Section 1.1 and defining the class “ViewNode”. These statements are the same for all objects of class “BaseNode”. Therefore, we could implement the answer to this type of questions once and for all and then use this implementation to answer the question for every particular object p of this class. So, actually, we need to have some method attached to class “BaseNode” and implementing the semantics of class “ViewNode”. The first proposal for such a method could be:

```
isInstanceOfViewClass() : Boolean;
```

This method returns true if the object must belong to the virtual view class defined **ON** the given base class and false otherwise. However, if we want to be able to define more than just one view class **ON** the same base class, we must be able to specify the exact view class the object must belong to. So we would like to change the method into a different one:

```
getClassName() : String;
```

Now, we can get the name of the view class the object must belong to even if there are more view classes defined **ON** the same base class. However, a question could arise – would not it be possible for some object to belong to several view classes simultaneously? We can imagine such a situation, for instance, in a case of creating a more detailed hierarchy in the view than it is in the base. Figure 2 shows such an example – we have some part of a possible flowchart metamodel as a base having only one class “Element” while two more classes – “FlowStart” and “FlowEnd” – are introduced in the view metamodel. Some objects of base class “Element” (e.g., an action) can therefore belong to both newly introduced view classes (and to their superclass as well, actually). We can say – a *1:n* mapping between base classes and view classes is possible in a general case.

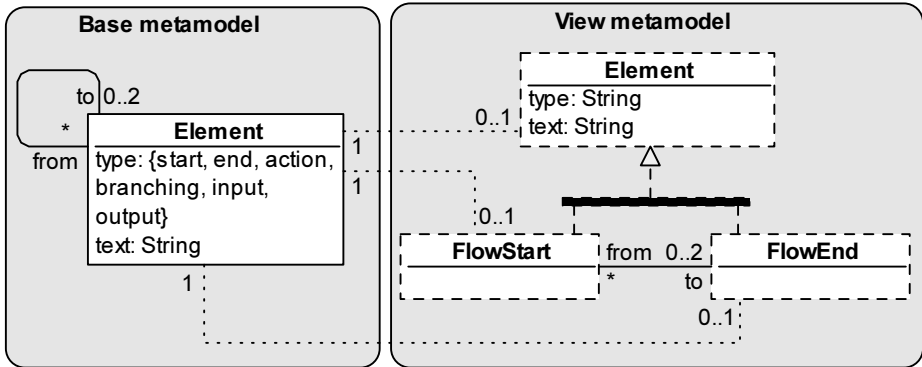


Figure 2. One base object can belong to several view classes at the same time.

Our *getClassName* method can not cope with this situation, so we must change it again:

```
isInstanceOfViewClass(viewClassName: String): Boolean;
```

Now, it must be clear that this method is universal enough to be used in view definition mechanism in general. We must implement this method for every base class **ON** which some view classes are defined. Likewise, other methods (with a previously known signature) must exist there in order to be able to traverse the “virtual” instance graph while actually working with real instances only. For example, we need to have a method for getting the value of some “virtual” attribute of the given object, a method for getting the object which is connected to the given object via some “virtual” link, etc. Also, other methods must exist for altering the instance graph (adding new objects belonging to some “virtual” class, etc.).

In this paper, only the graph traversal is taken into consideration when building views. However, the same view building mechanism can be adopted when developing a set of methods for the instance graph alteration. The complete set of methods for traversing instance graphs is listed below:

1. isInstanceOfViewClass (viewClassName: String): Boolean;

The method returns true iff the instance it is called on must virtually belong to the given class of the view metamodel (the class *viewClassName*). For example, in the case of our oriented graphs, this method (like any other of the

following ones) would be defined for classes “BaseNode” and “BaseEdge”. In the case of “BaseEdge”, this method would return true iff it is called with parameter “ViewEdge” and the value of the attribute “isVisible” is equal to *true* for that instance. In the case of “BaseNode”, the behavior of the method would be even simpler – it would always return true when called with parameter “ViewNode” (and *false* otherwise) since we wanted to look at all nodes in our view.

2. getViewAttrValue (viewAttrName: String): String;
The method returns the value of the view attribute *viewAttrName* for the object it is called on. Again, in our example – when this method is defined for the class “BaseEdge”, it must return the concatenation of *value* and *unit* of the instance attached to the instance this method is called on when called with a parameter “label”. In the case of “BaseNode”, this method must return the value of the base object attribute “caption” when called with a parameter “name”.
3. getFirstBaseObjectByViewLink (viewRoleName: String): Thing;
The method returns an instance of a base class that have to be reachable from the instance the method is called on by the role *viewRoleName* when called for the first time. Here and further, the type “Thing” must be understood as a superclass of all the classes of the base metamodel, so the actual class of the returned object does not have to be known before.
4. getNextBaseObjectByViewLink (prevBaseObject: Thing, viewRoleName: String): Thing;
The method returns the next instance according to rules given in the precedent clause.
5. existsViewLinkToBaseObject (viewRoleName: String, baseObject: Thing): Boolean;
The method returns *true* iff there must exist a link between an instance the method is called on and *baseObject* with a role name *viewRoleName* at the side of *baseObject*.

Indeed, having these methods, we can simulate the traversal of the view instance graph by traversing the base instance graph and calling these methods when needed. For example, if we want to start the traversal by getting the first (“virtual”) instance of class “ViewNode”, we must write something like this (written in pseudocode where the exclamation mark means the NOT operation):

```
n = getFirstObject ("BaseNode")
while (!n.isInstanceOfViewClass ("ViewNode"))
    n = getNextObject ("BaseNode", n)
```

It is assumed here that there exist some commands in the model transformation language in use for getting the first object of a class and for getting the next object of a class. To distinguish the view definition methods from the ordinary commands of the model transformation language, the former ones (only one method in this case) are underlined. Now, how can we be sure which base class must be inspected in those commands? Here, we must remember the very important property of every view mentioned previously – every view class is defined **ON** exactly one base class –, so this knowledge (function **ON** from the set of view classes to the set of base classes) is a part

of the view definition (the other part is the implementation of abovementioned methods). Since the derivation of the actual base class does not involve human attendance (it can be obtained using the function `ON`), it can be done automatically. That is, a programmer could write the program thinking only in terms of the view metamodel and using the usual model transformation commands as if the metamodel really existed there:

```
n = getFirstObject ("ViewNode")
```

Then, an automatic compilation could take place and the previously mentioned code could be generated. So, programmers would work with the view metamodel as if it really existed in the real world (in repository) while their programs would still be able to work when compiled to ones working with the real existing base metamodel. The next section demonstrates the compilation process in more details.

2. Making Concepts of View the Only Visible Ones

As it was said above, the goal of developing the mechanism for building views is to let programmers write their programs keeping in mind only the view metamodel designed for specific purposes. A universal compiler would then take the program working with the view and transform it to a program working with the base.

Basic approach of this step is depicted in Figure 3. Here, the compiler is specified as a model transformation program transforming models of the view metamodel (MM_V) to models of the base metamodel (MM_B). This classifies the compiler as a higher order transformation (HOT [8]). Apart from MM_V , the person writing the compiler needs to be aware of base class methods as well. Hence, sets of M_T (traversal methods) and M_A (altering methods) also go as input data of the compiler specification. It must be noticed that only signatures (and the semantics, of course) of these methods are needed here. The compiler specification does not need to know the implementation of the methods. Therefore, once specified, methods can be re-implemented without changing the compiler.

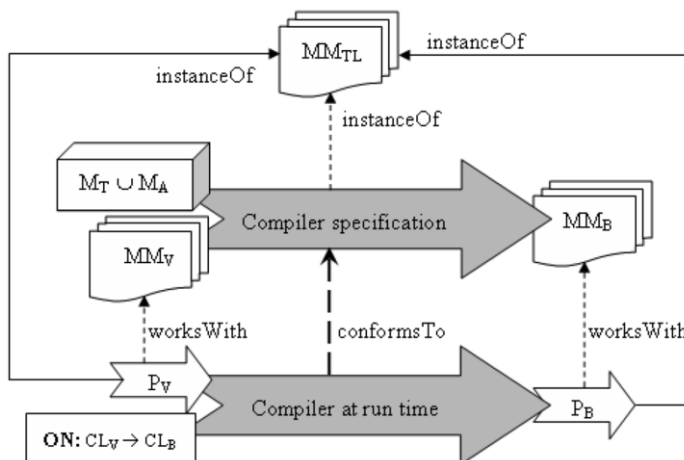


Figure 3. The compilation schema.

At runtime, compiler takes some program P_V working with the view metamodel and transforms it to a program P_B working with the base metamodel. Additionally, the function **ON** from view classes to base classes is taken as input data as well. Finally, there is also a metamodel of the transformation language (MM_{TL}) used to write the model transformations. It is assumed here that compiler is written in the same language as input and output transformations are written in. However, this is certainly not a prerequisite – other model transformation language can be used as well.

Let us see an example of a program working with the view in order to make it clearer. Let us suppose we want to show a message box for each edge containing the name of the node the edge starts in. A program P_V written in pseudo-code together with its translation to the program P_B working with the base metamodel is seen in Table 1. As said before, the base class methods are underlined here for better distinction from the transformation language commands.

Table 1. An example program for the view and for the base.

Program for the view (P_V)	Program for the base (P_B)
<code>e = getObject ("ViewEdge");</code>	<code>e = getObject ("BaseEdge");</code> <code>while (!e.isInstanceOfViewClass ("ViewEdge"))</code> <code> e = getNextObject (e, "BaseEdge");</code>
<code>label start;</code>	<code>label start;</code>
<code>if (e=NULL) goto done;</code>	<code>if (e=NULL) goto done;</code>
<code>n = getLinkedObject (e, "v_start");</code>	<code>n=e.<u>getFirstBaseObjectByViewLink</u>("v_start");</code>
<code>s = n.name;</code>	<code>s = n.<u>getViewAttrValue</u> ("name");</code>
<code>showMessage (s);</code>	<code>showMessage (s);</code>
<code>e = getNextObject (e, "ViewEdge")</code>	<code>e = getNextObject (e, "BaseEdge");</code> <code>while (!e.isInstanceOfViewClass ("ViewEdge"))</code> <code> e = getNextObject (e, "BaseEdge");</code>
<code>goto start;</code>	<code>goto start;</code>
<code>label done;</code>	<code>label done;</code>

We can see in Table 1 the general scheme of the way newly introduced methods are used in the program working with the base metamodel. Some details are perhaps left out of this general scheme. For example, what happens if there are no instances of the class “BaseEdge” belonging to the class “ViewEdge”? In the program P_V , the third row will tell us to go to *done* and the program will end. At the same time, in the program P_B , it seems like there would be an infinite loop in the first row. Since we do not know the implementation of the view definition method “isInstanceOfViewClass” for the class “BaseEdge”, we can not actually know it for sure whether or not the loop will end. For example, it can be said in the mentioned method that *NULL* value is a perfectly fine instance of a class “ViewEdge”. If not so, the generated P_B code must perhaps be modified slightly. However, such details are left out here for better perceptibility.

The compiler has generated the program P_B knowing only signatures and meaning of the methods. Now, it is the responsibility of the implementation of these methods to assure that the semantics of both transformation programs (P_V and P_B) are the same. In order to give a more detailed insight, a specification of some method must be presented. For example, the method “isInstanceOfViewClass” for the class “BaseEdge” can look like this (in pseudo-code):

```

function BaseEdge::isInstanceOfViewClass(viewClassName: String): Boolean
{
    if (viewClassName = "ViewEdge" AND this.isVisible)
        return true;
    else
        return false;
}

```

Since there is only one view class (namely, “ViewEdge”) defined **ON** the base class “BaseEdge”, the method can only return true when called with a parameter “ViewEdge” (and also – only if the edge is visible, according to the statements defined in Section 1.1).

3. Demonstration of the Approach

In order for us to be more certain of success, the view mechanism had to be implemented and tested for different cases. A model transformation language L0 [9] has been used to demonstrate the approach of making views and working with them. It must be noticed that the approach is not confined to any particular transformation or programming language, so any other language including MOF QVT [10] can be used instead.

L0 is a procedural language consisting of simple commands like “addObj”, “addLink”, “deleteObj”, “deleteLink”, “attr”, “setAttr”, “first”, “next”, “link”, “label”, “goto” and others (see [11] for more details). It has efficient compilers to C++ and Java for different model repositories (MiiRep [12], JGraLab [13], EMF [14]). The reason of choosing the language L0 is that it serves as a base language to which some other higher-level model transformation languages L1 to L3 [15] are compiled. Language L3 in its turn serves as a base language to which a very high-level graphical model transformation language MOLA [16] is compiled. Thus, by introducing the view compiler into the language L0, it suddenly becomes possible to work with views using any of these higher-level languages.

Figure 4 demonstrates the “MOLA to C++” compilation schema. The compiler compiling programs working with the view to programs working with the base is written only for the language L0. At the same time, the existence of such a compiler of L0 programs lets us automatically write our programs working with views in all the languages mentioned above. The bootstrapping process will do the job [17].

A transformation program making the step across the border between the view and the base (the view compiler) has been itself implemented in L0. In the first version, the compiler is aware only of model traversal methods presented in Section 1.2. In the next version, model altering methods will be added as well.

An example from Section 1 has been used to define a view. The function **ON** has been defined in the way seen in Figure 1. Since only graph traversal has been considered in this first demonstration, only the five traversal methods outlined in Section 1.2 had to be implemented for every base class **ON** which a view class is defined. So, the total of ten methods (there are two “interesting” base classes – “BaseNode” and “BaseEdge”) has been implemented in L0 (it is possible to define operations of classes in L0).

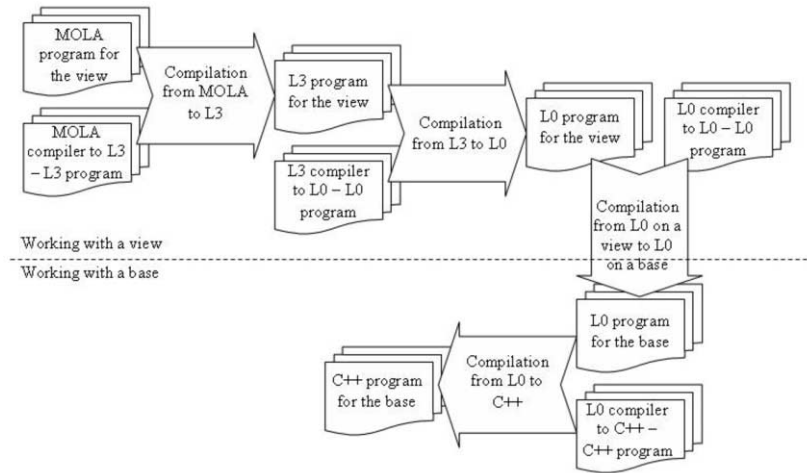


Figure 4. The “MOLA to C++” compilation schema.

Finally, several model transformations have been written to test the newly developed view. The transformation writer had not been aware of the base metamodel really existing in the repository and used only the view metamodel when writing the transformations. Test approved that the view mechanism is really usable in the case of graph (that is, model) traversal algorithms. Also, there are currently no covert threats seen regarding to the implementation of other methods responsible for graph altering and to adapting the view compiler accordingly.

4. Use Cases of the View Mechanism

The main use cases of the view mechanism outlined in the paper is in occasions there are one very big metamodel and several users want to understand different parts of the metamodel. Like mentioned in introduction, this is a very common situation in the field of graphical tool building. The metamodel storing the data of some tools can grow very big as increases the set of various features the tool must assure.

For example, in the Transformation-Driven Architecture [18], there are several interfaces each of them consisting of an engine and a metamodel that engine understands. It may seem like a very clear division, and that is really so in a logical level. However, all the metamodels are still physically stored together in one repository. So, there is actually one very big metamodel to which different users (e.g., different engines) try to look taking in mind only their specific needs.

This is a very simple use of a concept of a view – to take only a subset of a base metamodel to be a view metamodel. In this case, methods to be written for the base classes would be very trivial, since the base and the view classes are actually the same. Moreover, if the view is a real sub-case of the base and no other statements must be fulfilled, there is no need for the methods at all – programs written for the view metamodel will automatically work with the base metamodel as well. The compilation step does not need to be performed in this case.

More interesting situations can arise when trying to look at the same logical data stored in different formats (that is, in different metamodels). The most natural example here is perhaps a “domain – presentation” paradigm. The data are stored in some domain metamodel forming the logical structure while the graphical engine works with some presentation metamodel consisting of graphical primitives (like the ones in [19]). Now, the implementation usually involves the definition of some sort of mapping relation between those two metamodels. Regardless of the type of mapping (either static or dynamic using model transformations), both metamodels together with their instances are kept physically in the memory. The data redundancy is being introduced that way. This is the part where the notion of a view can come into play. One of the metamodels can be chosen to be “the real one” forming the base while the other would play the role of a view. The instance forming the tool would then be stored in only one exemplar while model transformations could still “think” of both metamodels and both models as of really existing ones.

5. Related Work

Let us look back at the example outlined in the introduction. One can need to define a derived class “Good Student” from the class “Student” of the university ontology saying that a student belongs to the class “Good Student” iff he has collected certain amount of credit points and passed all the mandatory courses. Since the statement contains some arithmetic, it can not be described by OWL (Web Ontology Language [20]) restriction classes (unless you make a special way of representing points and grades). The OCL (Object Constraint Language [21]) offers some mechanisms for defining derived classes, but they are very limited in many cases (e.g., no new attributes can be added to the derived class). A solution using views, on the contrary, is very convenient for defining such derived classes. However, despite the importance of the problem, no significant papers presenting any threads in exactly this direction can be found. Though the term “view” itself is used quite often (e.g., it is one of the basic concepts of MOF QVT [10]), its meaning is often quite different from the one used in this paper.

Somewhat related to the concept of a view is the notion of model type in the Kermeta metamodeling language [22]. For example, J. Steel and J. M. Jezequel define the model type as its metamodel [23]. The refinement of the model type is presented by the same authors in [24]. The main problem here is how to find a metamodel in the repository to which the given model conforms. Their desire is the same as presented in this paper – to make model transformations to be able to work with the given model. The problem of safe metamodel replaceability is being addressed here and an algorithm of model type conformance is provided. That is, we have some type for the given model and we want to check whether it satisfies the required model type or not.

The same problem of model subtyping is further analysed in the Maude language [25]. Maude is a high-level language and a high-performance interpreter and compiler in the OBJ algebraic specification family that supports membership equational logic and rewriting logic specification and programming of systems [25]. Here, the notion of metamodel subtyping (the fact that metamodel M' is a subtype of metamodel M , or $M' \leq M$) is defined in precise logic expressions. The subtype of a metamodel is further defined through the concept of a subtype of several metamodel elements – packages, classes, attributes and references –, which in their turn are then reduced to a \leq relation

within the primitive datatypes like Integer, Double and Boolean. Then, the assumption is made that there is no single type for a model. To find some model type, we can start with an initial metamodel of the given model (the metamodel with the minimum set of elements that is a valid metamodel for the model) and to traverse the metamodels in the given repository looking for metamodels which can safely replace that metamodel using the provided subtyping algorithm. Maude uses KM3, a specialized textual language for specifying metamodels [26], to illustrate the approach.

However, the main problem with this approach is that the provided metamodel must be very similar to the requested one. In fact, since it must be a real subtype of the requested metamodel, it is about the case mentioned in Section 4. The biggest problem is perhaps the necessity for equality of metamodel element names, e.g., classes (and other elements) must be named exactly the same in both metamodels in order for one metamodel to be able to be a subtype of the other. This is not requested in the concept of a view presented in this paper providing a way of making the view metamodels more flexibly.

The notion of model type is used also in MOF QVT. It is defined there by a metamodel, a conformance kind and an optional set of OCL constraint expressions [10]. The metamodel declared at the model transformation declaration level is called an effective metamodel, while the actual metamodel represents the metamodel that is used when executing the transformation. Some conformance between these two metamodels is then required. If the conformance type indicated in the model type definition is *strict*, the model must be a real instance of the effective metamodel. On the other hand, if the conformance type is *effective*, any instance in the model extent that has a type that is referred in the model type metamodel need at least to contain the properties defined in the effective metamodel metaclass and have compatible types. However, the binding between metamodel elements is based on name comparison here as well, and only some specific renamings using the *alias* tag are possible in order to introduce more flexibility. It is admitted that comparisons and classifications between model types are out of the scope of the MOF QVT specification.

The concept of a view metamodel is also present in [27]. Here, a view is defined as a description that represents a system from a particular perspective, and thus includes a subset of the system's elements (e.g., modules). This solution differs from ours in the aspect that they look at several parallel view metamodels instead of looking at one base metamodel having several view metamodels defined on it. Therefore, very important issues of model conformance and consistency checking between views need to be addressed here. We do not consider these issues because we have only one metamodel really existing in the repository.

A very similar-to-ours understanding of a view can be found in [28]. Here, also a term *virtual view* is used to denote a non-materialized view that is defined in a form of a view metamodel onto a really existing metamodel. The view is specified using a modified version of triple graph grammars (TGG). The definition of a view consists of a set of TGG rules allowing an arbitrarily complex structure of the real model to be generated when one wants to create some element of the view model. No model transformation is used to ensure the conformity of view and base classes' attribute values. Instead, OCL constraint annotations are used for that kind of purposes.

6. Conclusions and Future Work

In this paper, a method of defining views and allowing a transformation language to work with them was presented. One of the biggest benefits of using the proposed view definition mechanism was a possibility of avoiding the redundancy of data. Only a base metamodel was to be kept in the memory, while the view metamodel remained in the level of thoughts. Then, a compiler compiling a program working with the view metamodel to a program working with the base metamodel had been implemented as well, hence letting the end-user (the transformation writer) to operate with instances as if they were real instances of a real view metamodel. The proposed method among other things allows one to define a view on another view defined as outlined in the paper. From a meta-modeling point of view, this is a very beautiful characteristics of the view (as well as of modeling itself) being a sort of abstraction in respect to real systems.

In the paper, only some view definition facilities were discussed to give an overall insight into the view definition mechanism. The set of methods presented in Section 1.2 consists only of graph traversal methods. Wherewith, the current version of the compiler, compiling programs working with a view to programs working with a base, works only with the graph traversal algorithms. At the same time, it must be clear that extending the set of methods including the ones altering sets of instances (adding and deleting instances, and setting the values of attributes) does not require any changes in the view definition mechanism. When an appropriate set of graph altering methods is defined, the compiler would need to be supplemented to work with graph altering algorithms as well.

The other thing mentioned in the paper was that about the concept of **ON**. In the current implementation, **ON** is to be a pure function instead of an arbitrary binary relation. This means we are not able to define one view class **ON** several base classes. For example, we are currently not able to swap the base and view metamodels from Figure 2 and to look at the whole base hierarchy through only one view class “Element”. In some cases, however, a wish could arise of creating one view class for several base classes. For instance, one can need to define a view class as a union of some base classes. In order to turn **ON** into an arbitrary binary relation, we must introduce some extra knowledge in the view compiler, so it is still able to find the correct base instance in every situation. This would be a very beautiful feature of the view definition mechanism. Besides, that does not seem so hard from the implementation point of view – we do not have to change the other parts of the mechanism outside of the compiler.

The next thing to do would be to think of another (higher-level) view definition language being more user-friendly than the one presented in this paper (consisting of development of several methods for base classes). It would perhaps be a graphical language providing a possibility to define the semantics of these methods in a graphical configurator-like way, e.g., like the configurator of domain specific tools [29] used in the Transformation-Driven Architecture [18]. In case we could define such a language, another compilation step would arise – a view defined in that language would be compiled to the language described in this paper (we can call it the base language for defining views) and already having an implementation.

References

- [1] Meta Object Facility (MOF) Core Specification v2.0, OMG, document formal/06-01-01, 2006
- [2] MetaEdit+ Workbench User's Guide, Version 4.5, <http://www.metacase.com/support/45/manuals/mwb/Mw.html>, 2008
- [3] S. Kelly, J.-P. Tolvanen, *Domain-Specific Modeling: Enabling Full Code Generation*, Wiley, 2008, 448 pages
- [4] S. Cook, G. Jones, S. Kent, A. C. Wills. *Domain-Specific Development with Visual Studio DSL Tools*, Addison-Wesley, 2007
- [5] A. Shatalin, A. Tikhomirov. *Graphical Modeling Framework Architecture Overview*. Eclipse Modeling Symposium, 2006
- [6] A. Kalnins, O. Vilitis, E. Celms, E. Kalnina, A. Sostaks, J. Barzdins, *Building Tools by Model Transformations in Eclipse*. Proceedings of DSM'07 workshop of OOPSLA 2007, Montreal, Canada, Jyväskylä University Printing House, 2007, pp. 194 – 207
- [7] J. Barzdins, A. Zarins, K. Cerans, A. Kalnins, E. Rencis, L. Lace, R. Liepins, A. Sprogis, GrTP: Transformation Based Graphical Tool Building Platform, MODELS 2007, Workshop on Model Driven Development of Advanced User Interfaces, 2007
- [8] M. Tisi, F. Jouault, P. Fraternali, S. Ceri, J. Bezivin. On the Use of Higher-Order Model Transformations. *Lecture Notes in Computer Science*, Vol. 5562, Springer-Verlag, 2009, pp. 18 – 33
- [9] S. Rikacovs, The base transformation language L0+ and its implementation, *Scientific Papers, University of Latvia, "Computer Science and Information Technologies"*, 2008, pp. 75 – 102
- [10] MOF QVT Final Adopted Specification, OMG, document ptc/05-11-01, 2005
- [11] The Lx transformation language set home page, <http://Lx.mii.lu.lv>
- [12] J. Barzdins, G. Barzdins, R. Balodis, K. Cerans, A. Kalnins, M. Opmanis, K. Podnieks, *Towards Semantic Latvia. Communications of the 7th International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2006)*, Vilnius, 2006, pp. 203 – 218
- [13] S. Kahle. JGraLab: Konzeption, Entwurf und Implementierung einer Java-Klassenbibliothek für TGraphen, Diplomarbeit, University of Koblenz-Landau, Institute for Software Technology, 2006
- [14] Eclipse Modeling Framework (EMF, Eclipse Modeling subproject), <http://www.eclipse.org/emf>
- [15] E. Rencis, *Model Transformation Languages L1, L2, L3 and their Implementation*, *Scientific Papers, University of Latvia, "Computer Science and Information Technologies"*, 2008, pp. 103 – 139
- [16] A. Kalnins, J. Barzdins, E. Celms. *Model Transformation Language MOLA*, *Proceedings of MDAFA 2004, LNCS*, Vol. 3599, Springer-Verlag, 2005, pp. 62 – 76
- [17] J. Barzdins, A. Kalnins, E. Rencis, S. Rikacovs, *Model Transformation Languages and their Implementation by Bootstrapping Method. Pillars of Computer Science, Lecture Notes in Computer Science*, Vol. 4800, Springer-Verlag, 2008, pp. 130 – 145
- [18] J. Barzdins, S. Kozlovics, E. Rencis. *The Transformation-Driven Architecture*. Proceedings of DSM'08 Workshop of OOPSLA 2008, Nashville, USA, 2008, pp. 60 – 63
- [19] J. Barzdins, K. Cerans, S. Kozlovics, E. Rencis, A. Zarins. *A Graph Diagram Engine for the Transformation-Driven Architecture*. Proceedings of MDDAUI'09 Workshop of International Conference on Intelligent User Interfaces 2009, Sanibel Island, Florida, USA, 2009, pp. 29 – 32
- [20] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL>
- [21] UML 2.0 OCL Specification, OMG, document ptc/03-10-14, 2003
- [22] Kermeta, <http://www.kermeta.org>
- [23] J. Steel, J. M. Jezequel, *Mode Typing for Improving Reuse in Model-Driven Engineering. Model Driven Engineering languages and Systems, Lecture Notes in Computer Science*, Vol. 3713, Springer-Verlag, 2005, pp. 84 – 96
- [24] J. Steel, J. M. Jezequel, *On Model Typing*. *Journal of Software and Systems Modeling*, Vol. 6, No. 4, Springer Berlin, 2007, pp. 401 – 413
- [25] J. R. Romero, J. E. Rivera, F. Duran, A. Vallecillo, *Formal and Tool Support for Model Driven Engineering with Maude*. *Journal of Object Technology*, Vol. 6, No. 9, 2007, pp. 187 – 207
- [26] J. Bezivin, F. Jouault, KM3: a DSL for Metamodel Specification. Proceedings of the 8th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMODS 2006), Bologna, Italy, 2006, *Lecture Notes in Computer Science*, Vol. 4037, Springer-Verlag, 2006, pp. 171 – 185
- [27] R. F. Paige, P. J. Brooke, J. S. Ostroff, *Metamodel-based model conformance and multiview consistency checking*. *ACM Trans. Softw. Engin. Method*, 16, 3, Article 11 (July 2007), 49 pages
- [28] J. Jakob, A. Schürr, *View Creation of Meta Models by Using Modified Triple Graph Grammars*. *Electr. Notes Theor. Comput. Sci.* 211, 2008, pp. 181 – 190
- [29] A. Sprogis, *The Configurator in DSL Tool Building*, Accepted for publication in *Scientific Papers of University of Latvia, "Computer Science and Information Technologies"*, 2010

Associations as First-class Elements

Daniel BILDHAUER¹

dbildh@uni-koblenz.de

University of Koblenz-Landau

Abstract. Models in general and class diagrams as defined by the UML in particular play an important role throughout all steps of modern software development. However, the UML contains several modeling concepts which are not defined precisely and hence are used either rarely or in varying ways and with different semantics. In this article, an extension of n-ary associations as one of those concepts is presented including their refinement by specialization, subsetting and redefinition. DHHTGraphs are introduced as one realization of this extension treating associations and links as first-class objects. In particular, a metamodel and a possible implementation realizing association end refinement by association specialization are presented.

Keywords. N-ary associations, redefinition, subsetting, specialization, Hypergraph

Introduction

In modern software development processes, *models* are becoming the *central artifacts during all development phases* from requirements engineering to implementation and even maintenance and re-engineering. One of the most used kinds of models are *class diagrams* as defined by the UML [1] or in the variants defined by e.g. the metamodeling languages MOF [2] or EMF [3].

While the expressiveness of MOF and EMF is limited especially regarding associations, the UML provides several advanced modeling concepts such as n-ary associations and several possibilities to refine associations and association ends. While some of them, especially redefinition, are not defined precisely and thus are used in varying ways, others such as associations specialization and n-ary associations are rarely used at all. As stated by Genova et al. [4], n-ary associations are of limited use in the way they are currently defined in the UML. In practice, they are often simulated by relation-like objects.

Based on the work of Genova et al. [4] and the formal description of the semantics of redefinition in [5], this article tries to fill the existing gaps in the definition of these modeling concepts and presents a precise and concise definition of n-ary associations and their refinement by specialization, subsetting and redefinition. Furthermore, it shows how this semantics could be implemented treating links as first-class objects.

The next section presents an example use case, motivating the need of a precise definition of n-ary associations and their refinement. Section 2 introduces n-ary associations, the refinement concepts for associations and association ends. Furthermore, the exten-

¹This work is partially funded by the German Research Foundation (DFG), project number EB 119/6-1

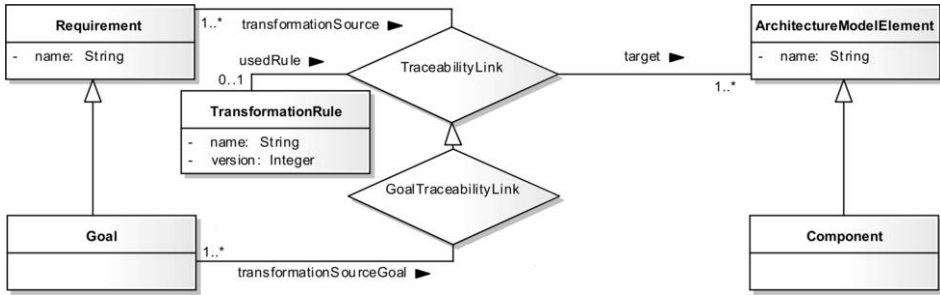


Figure 1. Example for a n-ary association

sion on the semantics of n-ary associations proposed by Genova et al. [4] is shown in this section with some further additions.

In Section 3 the refinement of n-ary associations and their ends is analyzed. A meta-model supporting the extensions of Genova as well as the refinement concepts is presented in Section 4, and a possible implementation is presented in Section 5. Other implementations, further work on refinement of associations and its ends and on n-ary associations are listed in Section 6. Finally, Section 7 summarizes the main propositions of this article.

1. Motivation

Being in modern model-centric development processes, *models* become the *central artifacts* in the software development lifecycle. Thus, their creation, modification and transformation by humans as well as by programs gets more important, while the generation of implementation code tends to be reduced to a primarily technical and automatable step. An appropriate modeling language should therefore be expressive, flexible and natural enough to allow for convenient and detailed modeling of systems. Simultaneously, the language's concepts need to be defined precisely for the language to be automatically processable by programs such as transformations.

Besides *domain specific languages (DSL)* in various variants, *class diagrams* as defined by the Unified Modeling Language (UML) are still one of the most used languages. While known to most modelers and easy to understand in their conceptional semantics and basic model elements, some of their details are defined either not precisely enough or in a way not obvious to all modelers. An example is shown in the following.

In the ReDSeeDS² project, model driven development has been combined with case-based reuse driven by requirements [6]. Software systems are developed using model transformation technologies during all development steps from requirements to code, where *traceability links* are used to record the transformations performed together with their affected elements. Besides only linking transformation sources with transformation targets, it may be useful to attach the used transformation rule as a further element to each traceability link. Figure 1 shows a simplified metamodel as an example for n-ary associations.

²www.redseeds.eu

Conceptually, the model describes that links of the n-ary association `TraceabilityLink` connect `Requirements` as sources with `ArchitectureModelElements` as targets. Additionally, if a link is a result of a transformation, the `TransformationRule` used should be attached to the link as a further element. `GoalTraceabilityLinks` are a special kind of `TraceabilityLinks`, as expressed by the association specialization. Furthermore, as defined by the association end `sourceGoal` and the additional redefinition constraint, `GoalTraceabilityLinks` may not connect to all kinds of `Requirement` but are restricted to `Goals`.

According to the semantics of n-ary associations defined by the UML, the model does in fact not represent this conceptual idea. As already mentioned by Genova et al. [4], the semantics of n-ary associations as defined by the UML is not very obvious and considers objects to be more important than links. It originates from the cardinalities in entity-relation diagrams, which allow to define the number of relations an element participates in by cardinalities in a flexible way, while no such possibility exists for relations. E.g., in UML as well as in ER diagrams every end of an association or relation denotes, that there is exactly one object connected to this end for each instance of the association while it is not possible to define, that for one end of a relation there may be more elements connected to one instance of the relation. In the example, it is not possible to define that at one `TraceabilityLink` there may be several `Requirements`. While such a preference of objects may be adequate in prescriptive models for code generation, where associations are only implemented indirectly by (coupled) attributes representing the association ends, such limitations seem to be no longer reasonable if models become the central software artifacts.

Besides the multiplicities, there is a further issue with the *specialization* of associations and the inheritance of association ends. Even if association specialization is rarely used explicitly and usually expressed by subsetting or redefinition of association ends, there is still a benefit of specialization as shown in Figure 1. Conceptually, the specialization of `TraceabilityLink` by `GoalTraceabilityLink` describes, that `GoalTraceabilityLink` is a special kind of `TraceabilityLink` which inherits all features from the superclass. While this semantics is compatible to the well-known and widely used specialization of classes and the descriptions of generalization in the UML Superstructure [1, p.72], it is not compatible to the description of associations in the same document [1, p.40].

Furthermore, as earlier shown in [5], the *semantics of redefinition and subsetting* and its interrelation to *association specialization* is not precisely defined. While a proposal for the refinement of binary associations has been given in [5], refinement of n-ary associations has not been tackled and is even more complicated. Probably, these complications are at least one reason, why n-ary associations are rarely used in practice but simulated by relation-like objects. While this workaround enables a more flexible way to define connections of such simulated association and classes on the one hand, it enforces a special treatment of those relation-like objects and their connected links in all algorithms working on the model. As shown later in this article, such a simulation is not necessary, if the concept of n-ary associations defined by the UML is extended and associations are treated as first-class objects. As a basis for this extensions, the single concepts are introduced in detail in the next section.

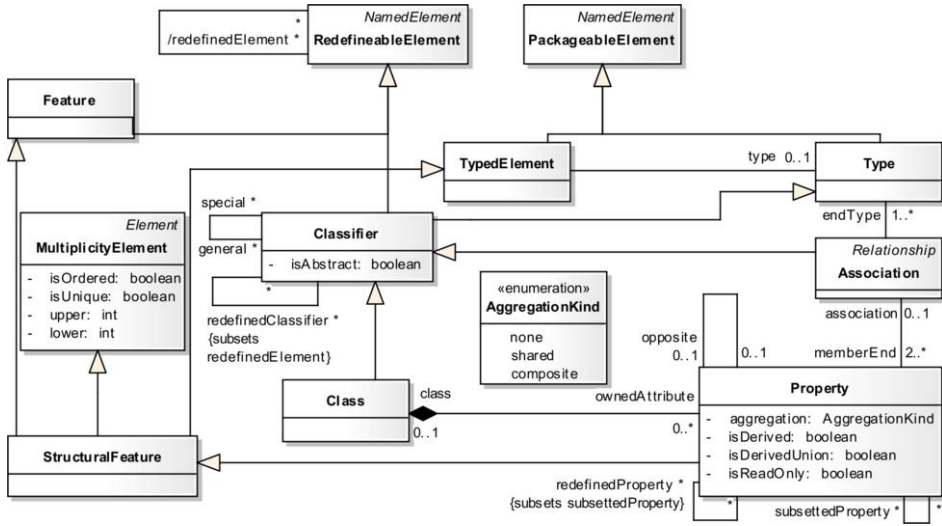


Figure 2. Relevant part of the UML metamodel

2. Definitions in the UML Metamodel

The UML is defined by a metamodel written in the Meta Object Facility (MOF) [2], which is a subset of UML class diagrams. In this metamodel, the UML concepts of classes, associations, association classes, and properties are defined by classes. Their relationships such as specialization of classes and associations as well as subsetting and redefinition of properties are expressed by associations between the relevant classes. An UML model is an instance of the metamodel. Figure 2 shows the relevant part of the UML metamodel with the classes `Class`, `Association` and `Property` and the relevant associations. In the following, the concepts are described in some more detail, using quotations of the UML specification.

2.1. Classes, Associations, and Properties

Classes as representations of entities of the modeled world are instances of the metaclass `Class`. According to the UML superstructure, “a class describes a set of objects that share the same specifications of features, constraints, and semantics.” The relations between entities of the modeled world are abstracted by *associations*, which are instances of the metaclass `Association`. The connections of classes and associations, also referred to as *association ends*, as well as attributes of classes and associations are instances of the metaclass `Property`.

As mentioned above, classes and associations can be specialized while properties can be refined by subsetting and redefinition. In the following, these concepts are explained using the model from Figure 1 as example. Furthermore, as n-ary associations have a special semantics shortly introduced above, they are also described in a bit more detail together with the extension proposed by Genova et al. [4].

2.2. Specialization

Classes as well as associations can be specialized and generalized, as indicated by the reflexive association at `Classifier` with the rolenames `special` and `general` in the metamodel. According to the UML Superstructure, the specialization of a classifier means that

"An instance of a specific Classifier is also an (indirect) instance of each of the general Classifiers. [...] features specified for instances of the general classifier are implicitly specified for instances of the specific classifier." [1, p. 55]

Since `Class` as well as `Association` are both subclasses of `Classifier`, this notion holds for both of them with some additional constraints imposed for associations:

"An association specializing another association has the same number of ends as the other association. [...] every end of the specific association corresponds to an end of the general association, and the specific end reaches the same type or a subtype of the more general end." [1, p. 40] "The existence of a link of a specializing association implies the existence of a link relating the same set of instances in a specialized association." [7, p. 113]

As described by those statements and conforming to the examples and usages of association specialization in the UML, associations may be specialized, but it is assumed that association ends are not inherited on specialization but need to be defined explicitly for each association. While for binary associations notated as solid lines connecting two classes the ends are defined implicitly anyway, the necessity to define the ends seem to be not reasonable for n-ary associations, as shown by the example in Figure 1. In this model, the explicit definition of two additional association ends at `GoalTraceabilityLink` would make the model mode complicated without any additional gain in precision. Therefore, an extension treating classes and associations equivalently regarding specialization is described later on in this article.

2.3. Subsetting

Similar to the specialization of classifiers, the refinement of properties is supported in the UML by a concept named *subsetting*. While its definition is scattered over various places in the Super- and Infrastructure, a good description is:

"An end of one association may be marked as a subset of an end of another in circumstances where [...] each of the set of types connected by the subsetting association conforms to a corresponding type connected by the subsetted association. In this case, given a set of specific instances for the other ends of both associations, the collection denoted by the subsetting end is fully included in the collection denoted by the subsetted end." [7, p. 113]

Figure 3 shows an example for subsetting. For all instances of `Goal`, the set `target` contains all elements of the set `targetComponent`. As shown in [5], *subsetting of one end of a binary association and specialization of the association mutually imply each other*. Regarding n-ary associations, this statement needs to be generalized with respect to subsetting between two ends at the same association, as shown in Section 3.

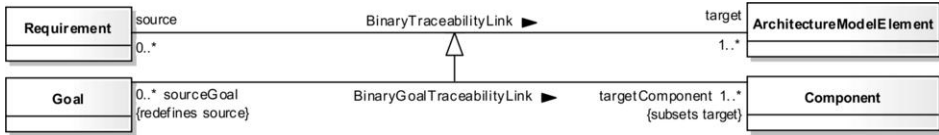


Figure 3. Subsetting and redefinition at a binary association

2.4. Redefinition

In addition to subsetting, the UML provides a concept named *redefinition* for the overriding refinement of properties as association ends and attributes. The redefinition concept is also used for the refinement of operations, but in this case not related to subsetting and association specialization and thus not considered here. While redefinition is not defined coherently in the UML Super- and Infrastructure, we assume the following semantics, which has been justified in detail in [5] and enables a practical usage of redefinition in our application domain.

"Redefinition is a relationship between features of classifiers within a specialization hierarchy. Redefinition may be used to change the definition of a feature, and thereby introduce a specialized classifier in place of the original featuring classifier, but this usage is incidental." [1, p. 41] "A redefined property must be inherited from a more general classifier containing the redefining property." [1, p. 128] "An end of one association may be marked to show it redefines an end of another in circumstances where [...] each of the set of types connected by the redefining association conforms to a corresponding type connected by the redefined association. In this case, given a set of specific instances for the other ends of both associations, the collections denoted by the redefining and redefined ends are the same." [7, p. 113]

As defined in this statements, redefinition is similar to subsetting with an additional constraint. In the context of the redefinition, all instances of the redefined property need to be also instances of the redefining one. All those instances are accessible using the redefining rolename as well as the redefined one. In Figure 3, the redefinition constraint at the association end `sourceGoal` specifies, that for all instances of `Component` the sets `source` and `sourceGoal` are identical. Thus, only `Goals` could be linked to `Components`, but they are accessible by the rolename `sourceGoal` as well as by the inherited one `source`.

2.5. N-ary Associations

While the associations usually used in class diagrams are binary with instances connecting exactly one source with exactly one target element, the UML allows for a more general kind of associations with a higher arity, whose instances connect more than two elements. The association `TraceabilityLink` in Figure 1 above is an example for such an *n-ary association*. The UML describes the semantics of an *n-ary association* as follows:

"For an association with *N* ends, choose any *N-1* ends and associate specific instances with those ends. Then the collection of links of the association that refer to these specific instances will identify a collection of instances at the other end. The multiplic-

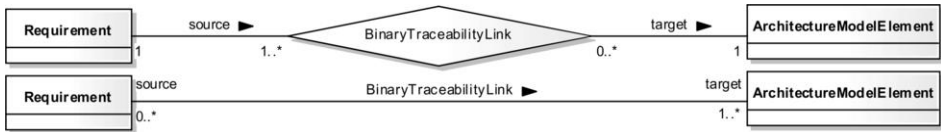


Figure 4. Binary association notated as a solid line and as a diamond

ity of the association end constrains the size of this collection." [1, p. 56] "For n-ary associations, the lower multiplicity of an end is typically 0. A lower multiplicity for an end of an n-ary association of 1 (or more) implies that one link (or more) must exist for every possible combination of values for the other ends." [1, p. 57]

Thus, the semantics of the association `TraceabilityLink` above is, that for each pair of `Requirement` and `ArchitectureModelElement` there is up to one `TransformationRule` linked to the pair and for each pair of `Requirement` [`ArchitectureModelElement`] and `TransformationRule`, there is at least one `ArchitectureModelElement` [`Requirement`] linked to the pair.

As shown by Genova et al. [4], this semantics does not allow for the detailed specification of participation constraints for the single classes connected by an n-ary association nor for the association itself. Genova et al. have proposed to extend the entity-relationship [8] like UML multiplicities by a second kind of multiplicities based on the method Merise [9]. This kind of multiplicity should be notated near the diamond depicting the association and is referred to as "inner multiplicity", while the Chen-like multiplicity of UML is called "outer multiplicity". In Figure 5, these two kinds of multiplicities are depicted at each association end. The semantics of those multiplicities is defined as follows: "The outer multiplicity [...] specifies the potential number of values at the specified end with respect to a combination of values at the other $n-1$ ends. A minimum outer multiplicity 0 means that the specified link end may be empty. The inner multiplicity [...] specifies the potential number of combinations of values at the other $n-1$ ends with respect to one value at the specified end. A minimum inner multiplicity 0 means that the elements at the end may not take part in the association." [4]

While these descriptions are still focused on the classes participating in an association and still considers objects to be more important than links, it is much easier to express the semantics of multiplicities if *associations* and its links are treated as *first-class objects*. In that case, the *inner multiplicities* define the *number of links* that may be connected to an object while the *outer multiplicities* defines the *number of objects connected to a link*. Assuming this semantics, the multiplicities at the association end named `source` in Figure 1 specify, that at each `Requirement` there may be an arbitrary number ($0..*$) of `TraceabilityLinks`, while at least one ($1..*$) `Requirement` participates in each such link. As the association is instantiated by a link, each end of the association is instantiated by zero, one or more ends of the link. Thus, each association end denotes sets of connections between one links and several objects and vice versa. The special case of a binary association and its representation as a solid line between the objects omitting the diamond and the multiplicities for the objects connected to a link is shown in Figure 4. Both notations in this figure are equivalent.

Similar to the set of connections at an object, which can be specified in more detail by constraints such as `ordered` or `unique`, also the set of connections at a link should be specifiable using the established constraints. Furthermore, the specialization of asso-

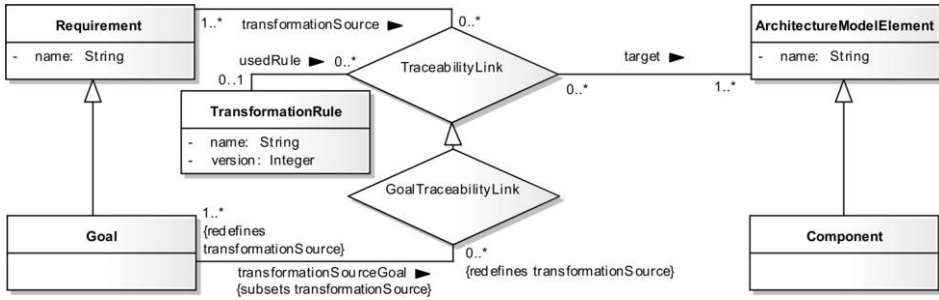


Figure 5. Model from Figure 1 extended by inner multiplicities and refinement

ciations and the refinement of association ends by subsetting and redefinition need to take this new semantics into account.

3. Refinement of N-ary Associations

As argued in Section 1 above, in a model-centric development environment there is no reason why the inheritance of association ends is treated different in the specialization of classes and associations. Treating associations as first-class objects, their properties such as their ends should be subject to specialization as they are for classes. Such an inheritance would allow a modeling as shown in Figure 5, where the association *GoalTraceabilityLink* specializes the association *TraceabilityLink* and inherits all its association ends. Thus, *GoalTraceabilityLinks* also have ends connecting to *ArchitectureModelElements* and *TransformationRules* without a need to specify these ends explicitly in the model. This semantics is analogical to the semantics of classes, which inherit the properties from their superclasses.

However, a refinement of a class or a association usually also includes a refinement of some of their properties such as attributed or connected association ends. While this refinement is implicit in the association notation of the UML, the concepts of subsetting and redefinition are used to refine the association ends connected to classes. The later can also be used for associations if an inheritance of their ends is assumed.

3.1. Subsetting

In the UML it is stated, that the ends of an association need to conform to the ends of every specialized association. In [5] it has been shown, that in fact the specialization of an association implies subsetting of all its ends. The semantics of subsetting described in Section 2.3 can be generalized to cover the extension of n-ary associations introduced above. A subset-relationship between two association ends denotes, that for all links of the association with the subsetting end the set of connections denoted by the subsetting end is fully included in the set of connections denoted by the subsetted end. Analogical, at each instance of the class connected to the subsetting end this restriction holds. As an example, the constraint *subsets transformationSource* at the association end *transformationSourceGoal* in Figure 5 defines, that for each instance of *Goal*, the set of *transformationSourceGoal*-connections is fully included in the set of *transformationSource*-ones. Similarly, this holds also

for `GoalTraceabilityLink`. It may be reasonable, not to restrict subsetting to two ends of an association and its specialization, but to allow also a subsetting relationship between two ends of the same association, similar to subsetting between two ends at the same class allowed in the UML. The semantics of subsetting is indeed not changed by this extension.

3.2. Redefinition

While subsetting can be used to define that one association end is a special case of another one, it does not restrict the occurrence of the specialized end. Such a restriction is possible by the usage of *redefinition*. As shortly introduced in Section 2.4, the redefinition of an association end restricts the existence of instances of the redefined end in the context of the redefinition. This enables a covariant specialization of properties and association ends, as it is possible to restrict inherited properties. Expressed in the terms of links connecting objects, it is possible to restrict the links at an object by redefining the other end of the association. The restriction of the objects connected to links is not done by redefinition as it is defined in the UML, but for each association it is specified explicitly which classes it connects without an inheritance of association ends on the specialization of an association. However, if the inheritance of association ends is assumed, the redefinition concept can be extended to restrict also the objects connected to links. An example is included in Figure 5. Similar to the multiplicities, the redefinition is notated separately for the association and the class at each end. While the inner redefinition constrains the links which may be connected to the objects of the class, the outer one restricts the objects which may be connected to the links.

One redefinition at an association ends implies also a subsetting of the end, since the restriction imposed by the redefinition is stronger than the subsetting. Thus, the statement in Section 2.4, that redefinition implies subsetting, is also true for the extended version of n-ary associations. However, we would propose to notate the subsetting explicitly for clarity.

4. Metamodel for N-ary Associations

As argued in Section 1, a precise definition is necessary for modeling concepts to be usable in practice. For the style of n-ary associations described above, such a definition and an appropriate implementation is given by the approach of *Distributed Hierarchical Hyper-TGraphs (DHHTGraphs)*. They are based on typed, attributed, ordered directed graphs, (*TGraphs* [10]), and extend them by means of distribution, hierarchy and hyperedges. In the context of this article, especially the hyperedges as a realization of n-ary links are relevant, while the other features are not considered in detail. In the context of DHHTGraphs, a slightly modified version of UML class diagrams called *Graph UML (grUML)* is used to describe graph schemas, which serve as metamodels for graphs. The diagram shown in Figure 5 is already a valid grUML diagram. The classes `Requirement`, `Goal` and so on denote *vertex types*, while the associations `TraceabilityLink` and `GoalTraceabilityLink` denote types of (hyper)edges connecting vertices. The language grUML is defined by a *metaschema*, the part relevant for the realization of n-ary associations is shown in Figure 6.

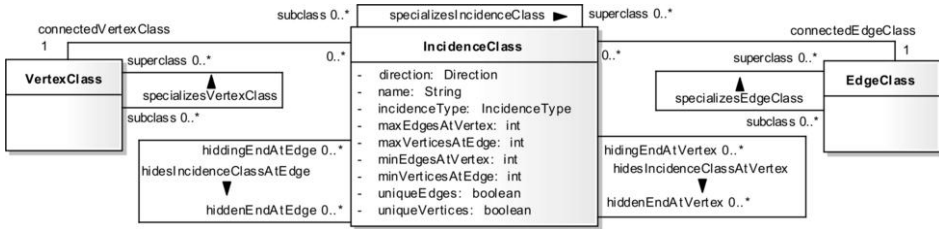


Figure 6. Part of the grUML-Metaschema

The connections points of nodes and their connecting edges are defined by *IncidenceClasses*, which are similar to properties in the UML. Each such one has a name, a direction and a type analogous to the class *Property* shown in Figure 2. Moreover, it stores the two kinds of multiplicities introduced in Section 2.5 which define the number of vertices connected by such an incidence to an edge and vice versa. Similar to the multiplicity semantics of the UML, these multiplicities affect the instances of the respective classes as well as all of their specializations. Furthermore, each *IncidenceClass* defines the uniqueness of the connections at a vertex or an edge by the attributes *uniqueEdges* and *uniqueVertices*. As in DHHTGraphs all incidences at a class as well as at a vertex are ordered by default, this information is not represented in the *IncidenceClasses*.

The possibility to specialize vertex-, edge- and incidence classes as representations of entities, relationships and their connection points is expressed by a relationship in each case. In every specialization, the properties are inherited. To allow a more detailed specification of connections, the two kinds of overriding refinement introduced in Section 3 are supported and expressed by the relationships *hidesIncidenceClassAtEdge* and *hidesIncidenceClassAtVertex*. These relationships represent the redefinition constraints as shown in Figure 5.

5. Implementation

The practical usability of a modeling language depends on frameworks which allow to handle model instances, e.g. by the automatic generation of source code which implements the structure and behavior described by the model. For a class diagram, every class in the model is transformed to a class in the desired implementation language, e.g. Java, and features such as generalization are realized by appropriate concepts of that language. For UML and MOF several such tools are established and used in practical software engineering. The possibly best established one is the Eclipse Modeling Framework (EMF) [3] based on the ECore metamodel. Similar to (Essential) MOF, ECore is restricted to binary associations and does not support associations as attributable and generalizable first-class elements, but reduces them to synchronized pairs of references representing the association ends. Each rolename is implemented by a collection containing the references to the appropriate linked objects. Since n-ary associations are not contained by ECore, there is no support by the code generation, but they need to be emulated by artificial classes. This realization of associations is common also for other modeling environments, even if some of them support subsetting and redefinition of association ends by synchronizing the relevant sets.

An alternative to the implementation of associations by their ends is the treatment of links as first-order elements as realized in the graph-library JGraLab [10] and also used for DHHTGraphs. Besides an easy realization of association attributes and generalization, this allows also a convenient implementation of subsetting and redefinition and especially of n-ary associations. If the associations defined in the model are implemented as classes with the links being instances of these classes, the role access can be realized by a traversal of the respective link objects. Moreover, as shown in [5], this implementation allows to use the specialization concept of the implementation language Java to realize subsetting and redefinition, since the traversal of the links for the subsetted or redefined rolename will include the links matched to the subsetting or redefining rolename without any need of further synchronization. Only at the creation of the links, some additional effort is necessary to deal with redefinition. In the listings below, the implementation of the model depicted in Figure 3 is shown as an example. As a special feature of JGraLab, the method `getThat()` allows to access the element at the other end of a link. The iteration over all `BinaryTraceabilityLink`-links to access the role `target` includes all `BinaryGoalTraceabilityLinks` which are created if a `Component` is added as a `targetComponent` to a `Goal`. The redefinition of the rolename `source` by `sourceGoal` is realized by an appropriate check as shown below.

```

class Requirement {
public BinaryTraceabilityLink addTarget(ArchitectureModelElement o) {
    return graph.createBinaryTraceabilityLink(this, o);
}

public List<? extends ArchitectureModelElement> getTarget() {
    List<ArchitectureModelElement> adjacences = new ArrayList<...>();
    BinaryTraceabilityLink edge = getFirstBinaryTraceabilityLink();
    while (edge != null) {
        adjacences.add((ArchitectureModelElement) edge.getThat());
        edge = edge.getNextBinaryTraceabilityLink();
    }
    return adjacences;
} }

class Component {
public BinaryTraceabilityLink addSource(Requirement r) {
    if (!r instanceof Goal)
        throw new SchemaException("Redefinition of role 'source' ... ");
    return graph.createBinaryGoalTraceabilityLink(r, this);
} }

```

Listing 1 Example implementation of role access on binary links

For n-ary associations, the implementation and in particular the role access is a bit more complicated. At first, as not only two elements but combinations of sets of elements are related by a link, there is the issue if such a role access is possible if only one element participating in a link is given. In the UML, ends of n-ary associations are treated only as member ends of that associations and not as properties of the relevant classes. Due to the UML semantics of n-ary associations described in Section 2.5, an access to a role is possible only if a combination of elements at the other end is given.

As an example, assume an object `r` of the class `Requirement` from Figure 5. An access to the rolename `target` on `r` may be interpreted in two differ-

ent ways and thus lead to two different results. Following the UML semantics, an access to the role `target` is only valid, if a combination of `Requirement` and `TransformationRules` is given but not for a single `Requirement`. However, if the extended semantics of n-ary associations is assumed, it may be reasonable to access the set of `ArchitectureModelElements` related to the `Requirement` by `TraceabilityLinks`. An example implementation of this semantics is shown in the listing below for the classes `Requirement` and `TraceabilityLink`.

```

public TraceabilityLink addTarget(ArchitectureModelElement o) {
    TraceabilityLink t = (...) graph.createTraceabilityLink();
    t.addSource(this); t.addTarget(o);
    return t;
}

public List<? extends ArchitectureModelElement> getTarget() {
    List<ArchitectureModelElement> adjacences = new ArrayList<...>();
    TraceabilityLink edge = getFirstTraceabilityLink();
    while (edge != null) {
        adjacences.addAll(edge.getTarget());
        edge = edge.getNextTraceabilityLink();    }
    return adjacences;
}

```

Listing 2 Implementation of role access on n-ary link in class `Requirement`

```

public void addTarget(ArchitectureModelElement o) {
    //create target incidence of IncidenceClass TraceabilityLink_target
    TraceabilityLink_target incidence = new TraceabilityLink_target(this, o);
    //add element to set of incidences at this edge and object o
    incidences.add(incidence); o.addIncidence(incidence);
}

public List<? extends ArchitectureModelElement> getTarget() {
    List<ArchitectureModelElement> adjacences = new ArrayList<...>();
    for (Incidence inc : incidences)
        if (inc instanceof TraceabilityLink_target)
            adjacences.add(inc.getVertex());
    return adjacences;
}

```

Listing 3 Implementation of role access on n-ary links in class `TraceabilityLink`

As shown in line two of Listing 2, adding an element to a set denoted by a role-name results in an edge to be created and the edge object returned by the create operation can be used to further attach elements to the relation represented by the edge. Special care needs to be spent on the refinement of association ends by subsetting between two ends at the same association. In general, refinement may be implemented either by synchronized sets, with all its advantages and disadvantages described in detail in the related work below, or by the usage of the specialization concept provided by the implementation language. In the latter case, each association end needs to be represented by a separate class and each link end by an appropriate object as realized in the examples above. While the main advantages of this implementation is the full support of the

model semantics, its main disadvantages is the increasing number of objects necessary to represent model instances. However, for the case of binary associations the approach has been proven in several projects which model instances containing several millions of objects and links and it can be assumed, that also for the case of n-ary associations the performance is sufficient. Furthermore, as shown by Scheidgen [11], all approaches based on coupled association ends and their synchronization are not able to reflect the semantics of subsetting and redefinition correctly without any further history.

The generated code allows to use the semantics of association specialization as well as of property subsetting or redefinition. The representation of links as edges which are first-order elements enables the usage of association attributes and thus offers modeling concept not present in other modeling frameworks like EMF. While in those modeling approaches the representation of relations carrying further information results in artificial objects and a special treatment of them in the algorithms, the TGraph approach keeps algorithms simple as it does not enforce such a distinction.

6. Related Work

Besides JGraLab and EMF there are some other modeling frameworks allowing to deal with model instances by code generation. However, the established ones are restricted to variants of EMF or EMOF without a support of n-ary associations. In the "MOF 2.0 for Java" implementation by Scheidgen [11], associations are represented by their ends which are realized by Java references stored in collections with subsetting and redefinition realized by synchronization of updates of those sets. Scheidgen shows, that this synchronization is not trivial, and proposes additional dependency information for each element to solve this issue. Scheidgen also discusses the alternative of implementing the links, but assumes them to be of limited use. In Section 5 we have shown, that this is not the case and that links can be used to implement subsetting and redefinition easily and without any need for additional dependency information.

A similar approach without the additional dependency information is presented by Amelunxen et al. [12]. As shown by Scheidgen [11, p 45], this approach has a slightly different update semantics where deletion of an element previously added to a set does not completely restore the previous state on all subsetted sets. The authors have also shown, that subsetting or redefinition of one association end implies subsetting of all other ends. However, in contrast to our interpretation of redefinition, which affects the association itself rendering it abstract in the context of the redefinition, only the classes are affected by the redefinition in their interpretation. Furthermore, the authors show, that `union` on an association end implies an abstract association and vice versa. This semantics is also supported by the implementation in JGraLab described above.

Olivé [13] identifies and formally defines different forms of refinement. He assumes the redefinition of a property (as representation of the refinement of a relationship participant) to be only a constraint on the connected elements of a relationship but independent of an association specialization [13, p. 230f]. While possibly useful when modeling without associations as first-class elements, this interpretation contradicts to the description in the UML superstructure [1, p. 18].

While several approaches for n-ary links or hyperedges have been developed, their usage and value has been discussed controversially. Engels and Schürr [14] propose to

keep links as simple as possible. They argue, that otherwise the separation of edges and vertices vanishes, leading to hyperedges pointing to hyperedges with their ends as new kind of plain edges. As shown above, the separation is actually assured by a uniform and consistent representation of simple edges and any more complex relationship.

7. Conclusion

If *models* become the *central artifacts* in the software development, the modeling languages used need to be expressive, flexible and natural enough to allow for convenient and detailed representation of entities as well as their relationships. However, even the widely used modeling language UML is not expressive enough for n-ary associations. In particular, besides the definition of multiplicities for n-ary associations, there are some issues regarding the refinement of associations and the inheritance of their properties. Based on Genova's [4] extensions to the multiplicities, also the refinement of associations can be improved if associations are treated as first-class objects. Using the established concepts of subsetting and redefinition of association ends in a slightly adopted way, a details specification of relationships is possible by n-ary associations. As a realization of this semantics, DHHTGraphs have been introduced and an exemplary implementation has been presented.

References

- [1] Object Management Group: Unified Modeling Language: Superstructure, Version 2.2. (February 2009)
- [2] Object Management Group: Meta Object Facility (MOF) Core Specification, Version 2.0. (01 2006)
- [3] Steinberg, D., et al.: EMF: Eclipse Modeling Framework 2.0. Addison-Wesley Professional (2009)
- [4] Génova, G., Lloréns, J., Martínez, P.: The meaning of multiplicity of n-ary associations in UML. *Software and System Modeling* 1(2) (2002) 86–97
- [5] Bildhauer, D.: On the relationship between subsetting, redefinition and association specialization. In: *Proc. of the 9th Baltic Conference on Databases and Information Systems 2010, Riga, Latvia (07 2010)*
- [6] Bildhauer, D., Horn, T., Ebert, J.: Similarity-driven software reuse. In: *CVSM '09: Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, Washington, DC, USA, IEEE Computer Society (2009)* 31–36
- [7] Object Management Group: Unified Modeling Language: Infrastructure, Version 2.2. (February 2009)
- [8] Chen, P.P.S.: The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.* 1(1) (1976) 9–36
- [9] Tardieu, H., Rochfeld, A., Coletti, R.: *La méthode MERISE, Tome 1: Principes et outils*. Les Editions d'Organisation, Paris, France (1985)
- [10] Ebert, J., Riediger, V., Winter, A.: Graph Technology in Reverse Engineering, The TGraph Approach. In Gimnich, R., Kaiser, U., Quante, J., Winter, A., eds.: *10th Workshop Software Reengineering (WSR 2008)*. Volume 126., Bonn, GI (2008) 67–81
- [11] Scheidgen, M.: *Description of Computer Languages Based on Object-Oriented Meta-Modelling*. PhD thesis, Humboldt Universität zu Berlin (October 2008)
- [12] Amelunxen, C., Schürr, A.: Vervollständigung des Constraint-basierten Assoziationskonzeptes von UML 2.0. In Mayr, H., Breu, R., eds.: *Proc. Modellierung 2006*. Volume P-82 of *Lecture Notes in Informatics*., Bonn, Gesellschaft für Informatik (2006) 163–172
- [13] Olivé, A.: *Conceptual Modeling of Information Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
- [14] Engels, G., Schuerr, A.: Encapsulated Hierarchical Graphs, Graph Types, and Meta Types. In: *SEG-RAGRA'95, Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation*. Volume 2., Elsevier (1995)

This page intentionally left blank

Semantic Technologies for Information Systems

This page intentionally left blank

Multilevel Data Repository for Ontological and Meta-modeling

Mārtiņš OPMANIS, Kārlis ČERĀNS

Institute of Mathematics and Computer Science of University of Latvia,

Rainis blvd. 29, Riga, LV 1459, Latvia

{Martins.Opmanis, Karlis.Cerans}@lumii.lv

Abstract. Metamodeling and Ontological modeling are two popular modeling approaches with their own technical tools, supporters and opponents. As a result of symbiosis of the two mentioned methodologies new metamodel is created and implemented in multilevel in-memory data repository. Repository implementation details are shown and several test- and use cases are analyzed.

Keywords: Class-based modeling, MOF metalevels, RDF data, data repository

Introduction

Activities like knowledge representation and modeling become more and more essential in a process of knowledge collection and presentation for definition, description and specification of computing system functions, architecture and design.

Classical MOF [1] four level approach is widely accepted in the world of modeling. However, in certain domains like modeling tool building there is need for working beyond metalevels, as defined by MOF. This can be illustrated by a simple example shown in Figure 1. In some domain there is clear distinction between elements of metadata (i.e. model) level (MOF M1), represented by *Domain Class* and data (i.e. object) level (MOF M0), represented by *Domain Object*. In usual data storages these levels are separated very clearly. In modeling tools, however, quite often a need arises to show elements from two metalevels at the same time, e.g. in a single diagram.

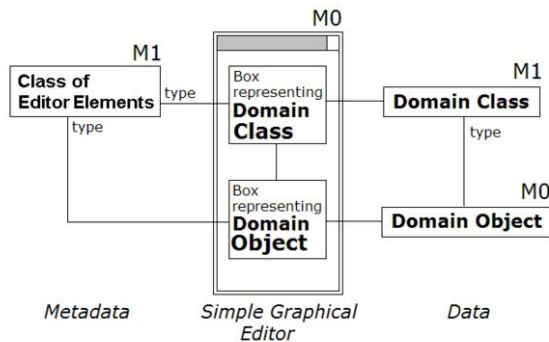


Figure 1. Metalevel clash problem in modeling tools

In Figure 1 in the middle a hypothetical *Simple Graphical Editor* window with two graphical elements – *Boxes* is shown. Each box is instance of metalevel element *Class of Editor Elements*. From the other hand, each box in editor represents some domain element (*Domain Class* or *Domain Object*) in data. The relations between elements in *Simple Graphical Editor* and *Data* are more than just informative since typical operations like “add new class” or “delete object” that are performed in the editor as a rule lead to essential changes also in domain data. The relation between the two shown *Boxes* (two elements of the same metalevel) in *Simple Graphical Editor* corresponds to *type* relation in *Data*, which is a relation between two elements of different metalevels. Therefore visually equivalent relations between editor and domain data elements are essentially different – relation between *Domain Object* and corresponding editor element is relation between two M0 level elements, in contrary to relation between *Domain Class* and corresponding editor element which connects M1 level element *Domain Class* with M0 level element *Box* representing *Domain Class* in editor.

This idea has been understood, e.g. in MetaEdit+ tool [2] where GOPRR modeling language has been adopted. GOPRR is not based on MOF and it uses different primary concepts: Graphs, Objects, Properties, Relationships and Roles.

In the Model-Driven Architecture (MDA) model transformations take important place. General model transformation task can be shown on Figure 2.

Let there be some modeling approach, whose static essence is described as MOF M2 concept *SourceMetamodel* and let *SourceModel* correspond to it. Let there be another MOF M2 concept *TargetMetamodel* and there be a need to transform *SourceModel* to *TargetModel* which conforms to *TargetMetamodel*.

The model level transformation depends on some transformation (or mapping) which translates the concepts of *SourceMetamodel* to the concepts of *TargetMetamodel*. In the classic model-to-model (M2M) transformation both metamodels conform to MOF metamodel (MOF level M3) what enables linking *SourceMetamodel* and *TargetMetamodel*.

If there is no such common metamodel, then further rising of metalevels is necessary till common basis *MMx* on some higher metalevel is found. If it is impossible to find such a basis then it will be impossible to transform one model to another.

Dashed frames around metamodel and corresponding model in Figure 2 are used to denote usual borders of one particular modeling tool, working with a single metamodel at a time.

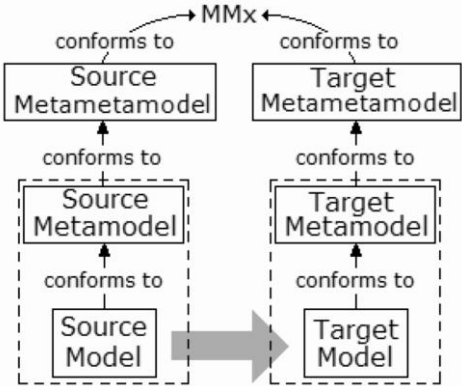


Figure 2. Model transformation task

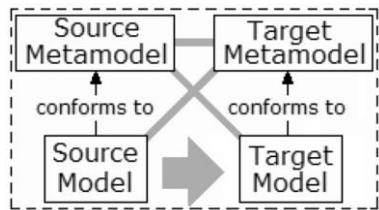


Figure 3. Proposed (JR) approach

In this paper we describe a solution to the mentioned problems using an originally developed multilevel data repository “JR”¹. Our approach shown in Figure 3 suggests the possibility of working with several metamodels (even more than two) and corresponding models at the same time in one working space. JR is based on JR metamodel (JRM), described in Chapter 2. Any metamodel which can be transformed to JRM, can be loaded in JR and all transformations can be done in JR. Therefore, JRM will work in a role of MOF allowing relatively simple transformation to it from other popular methodologies.

After loading in JR it will be possible to perform all transformations in JR, even beyond metalevels (denoted in Figure 3 by gray lines). For example, it will be possible to access directly *TargetMetamodel* content from elements of *SourceModel*, etc.

Working beyond metalevels in one working space is not a new idea. For example, it is realized in Visual and Precise Metamodeling (VPM) implemented in VIATRA2 (Visual Automated model TRAnsformations) framework [3]. VPM metamodel is given in Figure 4.

However, it seems that concepts used in the VPM metamodel are somewhat too general and serious application level must be built to successfully use models based on VPM metamodel. The scaling issues of the VPM metamodel implementation also are unclear.

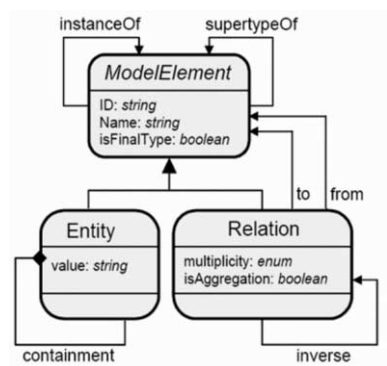


Figure 4. VPM metamodel (from [4])

¹ abbreviation stands for „New repository” („Jaunais repozitorijs” in Latvian)

Since the models and knowledge bases tend to become very large and the structural complexity of the conceptual part of these systems grows as well, we propose here a repository solution that is practically able to handle the amount of information corresponding to tens of millions of individual resources.

Amount of available RAM in contemporary computers allows loading model data in the main memory for quite large models. We offer the implementation of JR that is possible to address 10^{12} bytes as a metamodel based in-memory repository. Advantages of large data stores that are capable of structuring the data in accordance with arbitrary user-defined metamodel (or domain ontology) are in details justified in [5].

In [6] there is described necessity together with raw experiment data keep also some supplementary data (in the article the term “metadata” is used) which may have an essential value in further research but whose storage using usual implementations (such as relational DB) is not simple. The proposed solution will allow also keeping such kind of supplementary data without essential limitations.

1. Metamodeling and Ontological Modeling

Two popular modeling approaches – Metamodeling and Ontological can be represented by corresponding sets of languages - Unified Modeling Language (UML) [7] and Web Ontology Language (OWL)[8][9].

Despite different viewpoints and approaches to modeling [10], there are lot of efforts to find common points in the both [11][12] and/or to translate models from one world to another [13][14].

There a lot of “more or less common features” [10] that are used by Metamodeling and Ontological approach. Key features are collected in the Table 1 (consolidated from [12] and [10]):

Table 1. Common features in Ontological(OWL) and Metamodeling(UML) approach

<i>Ontological(OWL)</i>	<i>Metamodeling(UML)</i>
ontology	package
class	
individual	instance
values	attribute values
resource	model element
property	binary association, ownedAttribute
class, property	N-ary association, association class
data type	
subclass, subproperty	subclass, generalization
oneOf	enumeration
domain, range	navigable, non-navigable
disjointWith, unionOf	disjoint, cover
minCardinality, maxCardinality	multiplicity

If for specific modeling purposes it is enough to use only basic concepts that are similar in both approaches, then we deal with something that can be called “class-based modeling”. As the name denotes, *Class* is the main concept in this approach.

Also Object-oriented programming (OOP) is built on this concept. Differences in interpretation of class concept in various approaches are described in [15].

2. JR Metamodel (JRM) and Implementation

2.1. JRM Design

We let the class-based modeling that is common to both metamodeling and ontological modeling to be main modeling paradigm in the design of JR repository.

We include in JRM concepts from two metamodeling levels: MOF M1 (*Class*, *Association* and *Attribute*) and MOF M0 (*Object*, *AssocLink* (association link) and *AttrLink* (attribute link)).

Although having roots in Metamodeling, this design is in good correspondence also with Ontological modeling approach. In the Ontological modeling the concepts of *TBox* (terminological knowledge) and *ABox* (assertions) can be mapped to model and object level correspondingly.

To complete the JRM three more concepts are added: *DataType*, *DataValue* and *EnumValue*.

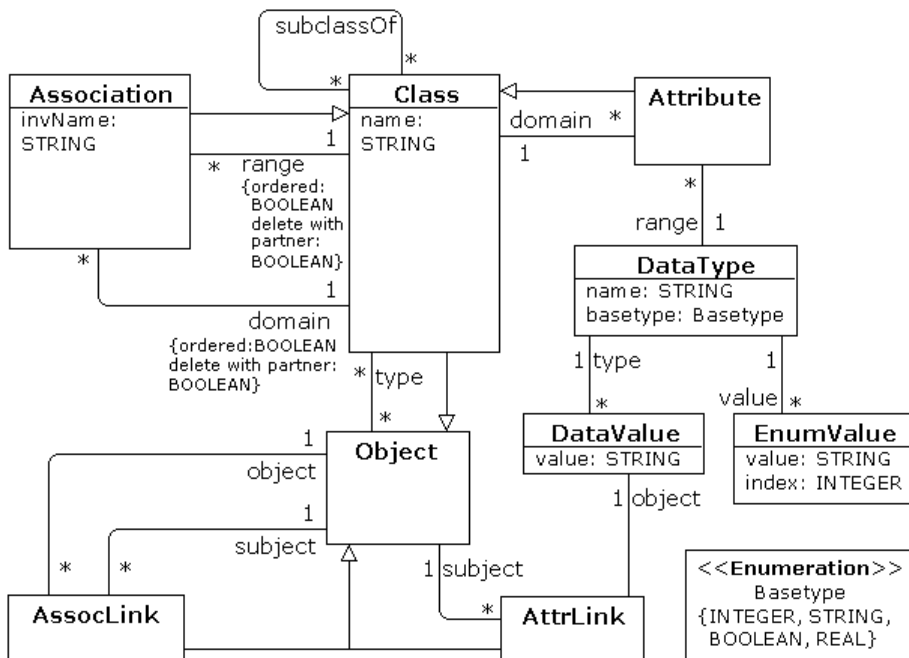


Figure 5. JR metamodel (JRM)

To simplify and shorten further description of models built on the JRM, any instance of JRM element will be referenced either by “*instance of <metamodel element>*” or by the same word as corresponding JRM element without capitalization (instance of *Class* in text can be referenced as *class*, instance of *Association* - as *association*, etc.). There may be also necessity to reference some entity of model without precise specification of corresponding JRM element. Such entities will be referenced as *things* (these “things” are similar to instances of *Thing* class in OWL; such a common superclass is not introduced in JRM).

Every *attribute* establishes a relationship between *class* and *datatype*, whereas *association* establishes a relationship among two *classes*. Instances of *Attribute* and *Association* will be accepted also as instances of *Class*. This feature allows building of more complex constructions, see, for instance later Figure 6.

Two special relationships between elements will be *type* and *subclassOf* relations. The *type* relation allows establishing type-instance (or *class-object*) relationship while the *subclassOf* relation is necessary for implementation of inheritance or generalization relationship between *classes*. Restrictions on usage of these relations are described in section 2.2.

The previously motivated need to work over metalevels will be met by allowing the instances of *Class* to be used as instances of *Object* without limitations.

Any strict two-layer model as a rule may be transformed into JR without essential effort simply by “forgetting” about subclass relation between *Class* and *Object*.

There is a built-in support of multiple inheritance in JR – the *type* relation between *Class* and *Object* is of cardinality “many-to-many”.

The resulting JRM is given in Figure 5.

Additional constraints to the JRM that are not shown in Figure 5 are:

- if *class* c_1 is in *subclassOf* relation with *class* c_2 and c_2 is in *subclassOf* relation with *class* c_3 , then c_1 is also in *subclassOf* relation with c_3 .
- if *class* c has *attribute* a , then also all *classes* having *subclassOf* relation with c have *attribute* a .
- if *class* c is *domain (range)* of *association* a , then also all *classes* having *subclassOf* relation with c can be used as *domain (range)* of a .
- if *type* of *object* o is *class* c , then *type* of o is also any *class* c_i where c is in *subclassOf* relation with c_i .
- if *domain (range)* of *association* a is defined with attribute *ordered=true*, then it is guaranteed that order of *assoclinks* having *type* a at its *subject (object)* end will not be changed and kept in creation order. If attribute *ordered=false*, then sequence of corresponding *assoclinks* is not defined and may change during model lifetime.
- if *domain (range)* of *association* a is defined with attribute *delete_with_partner=true*, then deletion of *thing* at *subject (object)* end of *assoclink* l having *type* a leads to deletion of *thing* at the opposite end of l .

There are several restrictions which are fulfilled at the moment of creation of a particular *thing* but can be violated by using particular JR functions, changing *type* and *subclassOf* relations among several *things*:

- *type* of *assoclink* is *association*,
- *type* of *attrlink* is *attribute*,
- if *association* is connected via *subclassOf* relation with *thing*, then this *thing* is *association*,

- if *attribute* is connected by *subclassOf* relation with *thing*, then this *thing* is *attribute*,
- if *datatype* is enumeration, then *datavalue* with *type datatype* must be *enumvalue* (one of this enumeration's values).

Several relationships can be named as “square relationships”. By dot(.) operation navigation via specified relation is denoted and also these relationships can be violated afterwards by explicit user actions:

- *assoclink.subject.type* must be the same as *assoclink.type.domain*,
- *assoclink.object.type* must be the same as *assoclink.type.range*,
- *attrlink.subject.type* must be the same as *attrlink.type.domain*,
- *attrlink.object.type* must be the same as *assoclink.type.range*.

2.2. Implementation Restrictions

During implementation JR some additional restrictions were implemented:

- the relationships consisting of chain of *type* and *subclassOf* relations cannot create loops. This restriction is considered quite natural in practical cases.
- despite no limitations to usage of equivalent names in general, there exists such a constraint for *datatypes*: no two different *datatypes* with the same name are allowed, as well as no two different *enumvalues* with the same string or integer value. It must be also pointed out that proposed approach is much more flexible if compared with MOF, where an essential constraint “ContentNamesMustNotCollide” exists [16].

There are constructions which are theoretically possible but which can not be created in JR. For example, an instance of *Association* may not have itself as a *domain* or a *range*. Impossibility to build such constructions is not a serious constraint for successful usage of JR. The possibility to use either “top-down” (when higher metalevel concepts are presented first and lower metalevel concepts follow) or “bottom-up” (when lower level concepts are followed by higher level concepts) or both approaches in a flexible manner is one of main features of JR repository. For every “reasonable” model there exists a sequence of API function calls allowing constructing such a model.

3. Working with JR

3.1. Mixed Metalevels Modeling

As it was mentioned before and as it can be seen from JRM, instances of *Association* and *Attribute* can serve as *Class* instances (see Figure 6).

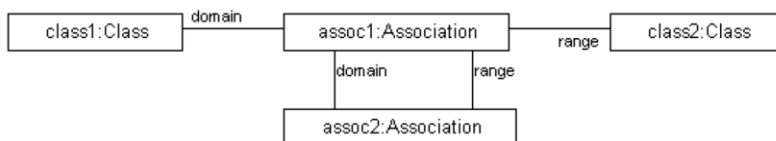


Figure 6. “Association instance (assoc1) as class” example

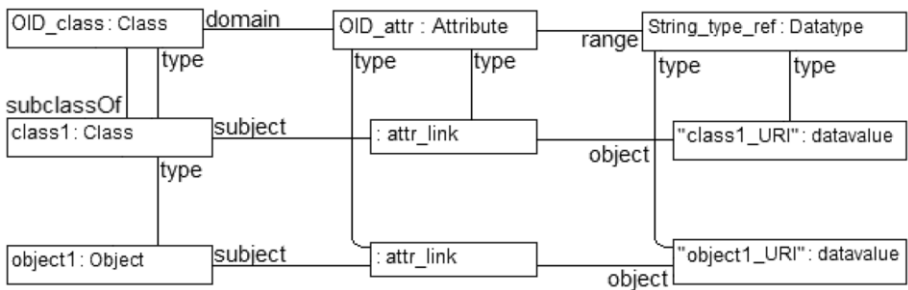


Figure 7. Simple mixed metalevels example

A simple example with mixed metalevels is given in Figure 7. This example reflects the situation where every single *thing* in the described model has its own “object identifier” (like unified resource identifier – URI in the Resource Description Framework(RDF)).

Let there be a *class* *OID_class* with an attribute *OID_attr* that describes the relation between *OID_class* instances and their respective object identifier values (as *datavalues* with type *String_type_ref*). Let *class1* be an instance of *OID_class* (denoted by the *type* relation from *class1* to *OID_Class*) and let *object1* be an instance of *class1*. The attribute *OID_attr* allows introducing a corresponding *attr_link* with type *OID_attr* for *class1*, however, in order to have such an *attr_link* for *object1*, there should be a *type* relation also from *object1* to *OID_class*. In our example we achieve this by letting *class1* be also *subclassOf* *OID_class* (an alternative possibility would be to define *OID_class* to be *type* for *object1* by a separate *type*-relation).

Now we can look at metalevels – *object1* is representative of M0. Obviously, *class1* is representative of M1 (type relation with *object1*). It is impossible to define exact metalevel of *OID_class*: it is representative of M2 (type relation from *class1*) as well as representative of M1 (subclassOf relation with *class1* denotes belonging to the same metalevel).

3.2. UML Models

Figure 8 contains a simplified variant of UML metamodel. JR can be used in UML-based modeling in at least two different ways.

The first way is to create JR instances for every construct in UML metamodel. The UML concepts from Figure 8 which are denoted as boxes (e.g. *Type*, *Class*, *Property*, *Association*, etc.) correspond to JR *Class* instances. The concepts which are denoted by links (e.g. *opposite*, *memberEnd*, *OwnedEnd*, etc.) correspond to JR *Association* instances. The concepts which are denoted in attribute notation (e.g. *name*, *default*, *isComposite*, etc.) correspond to JR *Attribute* instances. Note that the subclass notation from Figure 8, as well as the “part-of” constructs also can be directly modelled in JR. The instances of the UML metamodel itself are then related to the created “UML metamodel instances” by the *type* relation in JR.

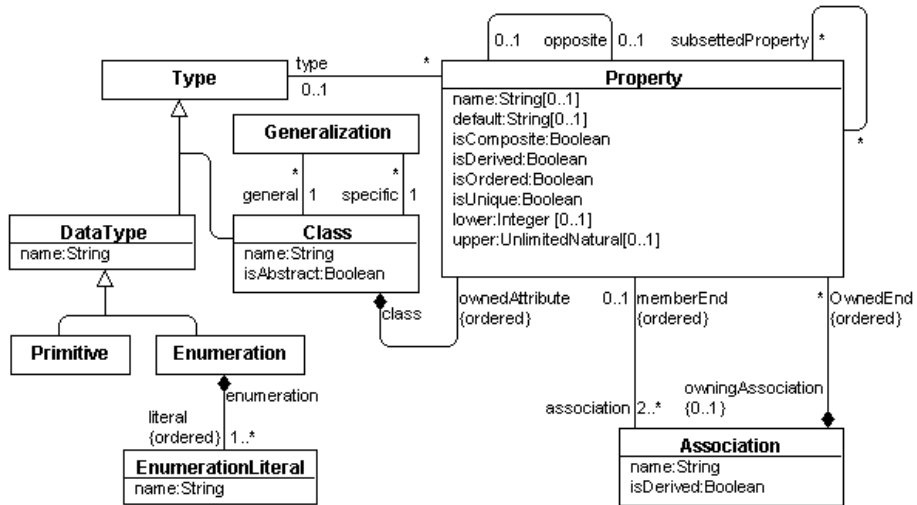


Figure 8. Simplified UML2.2 metamodel

The other way is to use a “deep embedding” of UML constructs into JRM by interpreting JR Class, Association and Attribute constructs as their UML counterparts and letting the UML metamodel instances be the instances of the corresponding JR construct. We note that this type of modeling, although it does restrict the possible UML constructs that can be used, can offer a higher efficiency due to relying on the optimisations that are built into JR.

3.3. OWL 2.0 Ontologies and Punning

In OWL there is notion of “punning”. Known also in previous versions, in OWL 2.0 there is given more attention to it [17]. Punning means that the same name can be used for description of two different entities assuming that these two entities describe the same thing from different viewpoints. One of simplest cases of punning is using class as instance. Since in JRM class is also an object, this case of punning can be modelled in JR in a direct way. It would be interesting to explore further the possibilities of JR that allow to regard a class as an instance within the OWL logical perspective.

3.4. JR Application Programming Interface

At the moment only available interface for working with JR is via application programming interface (API) function calls.

All API functions are grouped in a three groups: 1) functions for element creation and deletion, 2) functions for searching in repository, 3) other service functions.

The basic concept used in JR is REFERENCE that is somewhat similar to pointer concept in programming languages. REFERENCE is the only way to handle every element in repository and it contains information about element kind (i.e. *class*, *object*, *datatype*, etc.) which can be obtained through it. Dynamical REFERENCE type checking is provided always where only REFERENCE of special kind is acceptable.

This approach is very close to Dereferenceable Uniform Resource Identifier (Dereferenceable URI) concept used in Linked Data approach (next step in development of Semantic Web) [18] as well as pointers or "object references" in OOP.

Any search in JR is organized using *iterators*.

There are several different types of *iterators*:

- Iterators (7 types) for going through all *things* of one kind (i.e., through all associations, objects, classes, etc.)
- Iterators (28) for searching *things* directly connected to one specified *thing* in correspondence with metamodel (i.e. all *types* of particular *object*, all *objects* of particular *class*)
- Iterators (6) for searching *things* of special kind (having name) by name (i.e. all *classes* with name "MyName").
- Special iterators (4) for the following tasks:
 - From the given *object* and *attribute* find *data value* which is connected with the *object* by *attrlink* with *type attribute*,
 - From the given *data value* and *attribute* find *object* which is connected with the *data value* by *attrlink* with *type attribute*,
 - From the given *object*₁ and *association* find *object*₂ which is connected with the *object*₁ by *assoclink* as *subject* with *type association*,
 - From the given *object*₁ and *association* find *object*₂ which is connected with the *object*₁ by *assoclink* as *object* with *type association*.
- "Deep" iterators (6) for searching things taken in account subclass/superclass hierarchy defined by *subclassOf* relation. Creation of hierarchy of objects and classes given in Figure 9 can be done in any order – user may start either from subclasses (class31-class3-class0 – "bottom-up" approach) or from superclass (class0-class1 – "top-down" approach). Five objects in the given example will be traversed in the following order: o31,o311,o312,o11,o12.

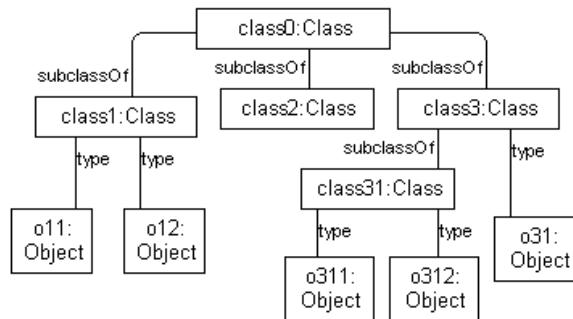


Figure 9. Class and object hierarchy example

4. JR Use- and Testcases

There have been two applications used for the repository testing. First of them – Molecular Phenotyping to Accelerate Genomic Epidemiology ("MolPAGE") project was chosen as a test bed for testing main capabilities of repository as well as working

with it through JAVA interface. The second one demonstrates capabilities of JR as one of key components in a migration process from relational databases to RDF databases.

4.1. MolPage

MolPage data repository was built on the basis of metamodel that is shown on Figure 10. Relations marked by {x} are subproperties of relation *persKinsman*.

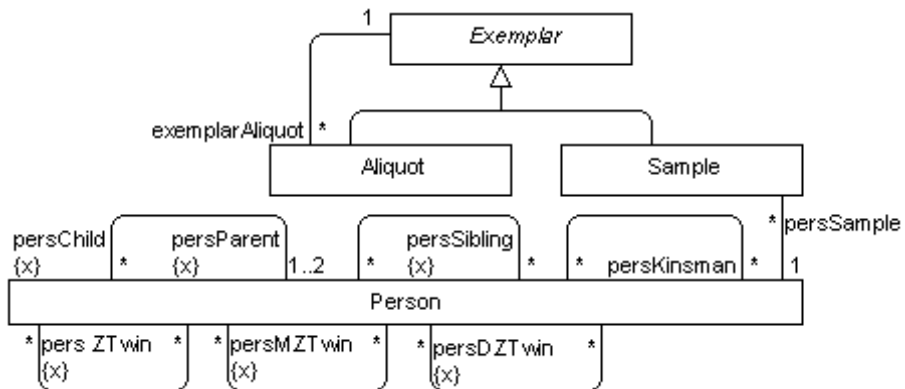


Figure 10. Simplified MolPage metamodel (without attributes and enumeration types)

The scale of this project can be characterized by number of instances: 5822 samples, 16225 aliquots and 2094 persons. Technically all data was given in 61 separate files (one with ontology description, 14 with description of samples, 29 with description of aliquots, 17 with description of persons).

In the case of this project it was natural to read the information corresponding to the model level first and the data afterwards. This work has been seriously simplified by three factors:

- the presence of ontology (there has been a more detailed diagram than the one shown on Figure10),
- the information that all ontology triples are collected in one file and
- the assumption that names in the model are unique.

It is necessary to say that in all projects considered here it has not been possible to import data without their previous investigation.

Conceptually data import process is shown in Figure 11: “1” denotes data and schema import from MolPage data given in *ntriples* format into SESAME data repository, “2” denotes import from SESAME repository into JR. Both imports are implemented as JAVA programs and therefore additional JAVA interface using Java Native Access (JNA) over plain C (in which JR is implemented) was necessary.



Figure 11. MolPAGE data processing

To import ontology data into JR it was necessary to:

1. Create “String” *datatype* with base type JR.STRING
2. Create data types corresponding to XMLSchema types GYEAR, DATE, FLOAT, STRING (these types were used in the given data)
3. Create supplementary class *_:ID_class* with attribute *OID* - the same what is given in RDF triples and afterwards will be used for identification of a particular *thing*
4. Automatically recognize (using *owl#DataRange*) enumerated types.
5. Get all classes (recognized using *OWL.CLASS*) from SESAME and create corresponding ones in JR.
6. Define every created class as subclass of supplementary class *_:ID_class*
7. Traverse all classes searching for superclasses. The assumption concerning uniqueness of class names was exploited here.
8. Loop through all classes searching for attributes and create the corresponding ones in JR. Possible attribute types are previously extracted enumerated types or type corresponding to XMLSchema built-in type. If it is discovered that *owl#DataRange* does not correspond to known data type, only correct situation is that this relation denotes association in the terms of JR. Before creating such association it must be checked that such association is not created in an opposite direction.
9. Loop through created associations and establish superclass relation between them if there is corresponding RDFS “subproperty of” kind relation in SESAME.

After these steps the MOLPAGE model layer was imported in JR. This step was followed by simpler data import step.

The following performance tests have been performed on Compaq nw8440 laptop with the following technical characteristics: HP Intel® Core™2 CPU T7400 @ 2.16GHz 994 MHz, 2,00GB of RAM.

Table 2. Execution times for MolPAGE queries

Test	Execution time in milliseconds	
	<i>JR</i>	<i>OpenLink Virtuoso</i>
G2	78	375
G5	<1	125
G6	578	17625

For the sake of comparison, the RDF triples have been loaded also into the OpenLink Virtuoso[19] datastore, where the corresponding tests (SPARQL[20] query in its original form) were executed, as well. For the JR case the code that corresponds to the original SPARQL query has been written in JAVA using JR API function calls. The comparisons of execution times are given in Table 2.

Despite the good JR performance test results, it was considered that data amount is too small to allow far-reaching conclusions from the obtained results. Also the

performance through JAVA interface was suggested to be worse if compared with plain C function calls.

4.2. Semantic Reengineering of Legacy Databases

One of applications where usage of JR has shown good results is export from relational databases to RDF databases [21].

Conceptually this process is shown in Figure 12.

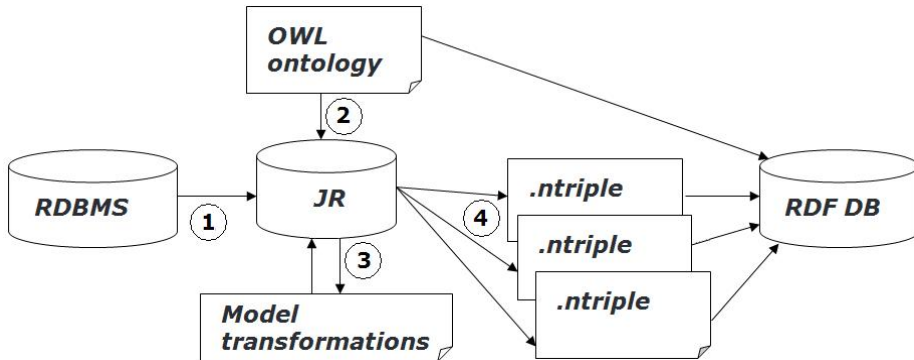


Figure 12. Export from RDBMS to RDF DB using JR (from [21])

During export process JR is involved and plays a key role in the following main steps (marked by numbers 1, 2, 3 and 4 in Figure 12):

1. The entity relationship (ER) schema and data from Relational Database Management System (RDBMS) is imported into JR,
2. The target OWL ontology is imported into JR,
3. Correspondence between ER schema and target ontology is established and necessary model and data transformations in JR are executed (the model transformations needs to be specified for each specific migration instance - a database schema and corresponding ontology)
4. Content of JR in specific native RDF format (“ntriples files”) for further import in RDF database is generated.

In practice this approach was used to migrate 6 Latvian medical registries into a RDF database. The source database contained about 100 tables and 1300 columns with $3 \cdot 10^6$ rows. As the result of export an ontology comprising 170 OWL classes, 200 OWL datatype properties and 800 OWL object properties has been filled with about $41 \cdot 10^6$ RDF triples.

The implemented relational database semantic reengineering task has shown that JR is capable of handling even very large data volumes. The experiments outlined also possibilities and directions for further improvement of JR performance.

5. Concluding Remarks

As it was shown, the proposed approach shows acceptable results in a number of practically important use cases. However, there is a long way to go till JR will become a widely used practical alternative for existing RDF data stores or model repositories. One direction to improve the usability of JR is in building appropriate application layer above existing JR API level in which there will be possible to execute SPARQL queries. The usage of JR in some practical applications has pointed out some parts and constructions of JR where performance can be improved. One such place is searching in cases where it is known that cardinality of a particular relation is “one-to-one” and therefore there is no need to iterate through *all* connected elements, but the only existing one can be obtained by a simpler construction.

References

- [1] Documents Associated with MOF Version 2.0, <http://www.omg.org/spec/MOF/2.0>
- [2] MetaEdit+, [http://www.metacase.com/support/45/manuals/Graphical Metamodeling.pdf](http://www.metacase.com/support/45/manuals/Graphical%20Metamodeling.pdf)
- [3] VIATRA2 framework, <http://dev.eclipse.org/viewcvs/indextech.cgi/gmt-home/subprojects/VIATRA2/index.html>
- [4] Balogh, A., Varro, D.: Advanced model transformation language constructs in the VIATRA2 framework. Symposium on Applied Computing Proceedings of the 2006 ACM symposium on Applied computing, Dijon, France, pp. 1280 – 1287, ISBN:1-59593-108-2 (2006)
- [5] Barzdins J., Barzdins G., Balodis R., Cerans K., Kalnins A., Opmanis M., Podnieks K.: Towards Semantic Latvia, Baltic DB&IS 2006, Communications, O.Vasileckas, J.Eder, A.Caplinskas (Eds). Vilnius: Technika, 2006. pp.203-218
- [6] Ailamaki A., Kantere V., Dash D.: Managing Scientific Data, Communications of the ACM, Vol.53, No.6, pp.68-78, 2010.
- [7] UML® Resource Page, <http://www.uml.org>
- [8] OWL Web Ontology Language Guide, <http://www.w3.org/TR/owl-guide/>
- [9] OWL 2 Web Ontology Language Document Overview, <http://www.w3.org/TR/owl2-overview/>
- [10] Hart L., Emery P., Colomb B., Raymond K., Taroporewalla S., Chang D., Ye Y. and Dutra M., Kendall E.: OWL Full and UML 2 compared, March 2004
- [11] Marrying Ontology and Software Technology, European Commission ICT research 7th Research Framework Programme project, <http://most-project.eu>
- [12] Pereira F.S., Staab S., Winter A.: On Marrying Ontological and Metamodeling Technical Spaces, ESEC/FSE'07, Dubrovnik, Croatia, pp. 439-448, ISBN:978-1-59593-812-1 (2007)
- [13] Brockmans S., Volz R., Eberhart A., Löffler P.: Visual Modeling of OWL DL Ontologies Using UML, S.A.McIlraith et al. (Eds.): ISWC 2004, LNCS 3298, pp.198-213, 2004
- [14] Brockmans S., Haase P., Hitzler P., Studer R.: A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies, Y.Sure and J.Domingue (Eds.): ESWC 2006, LNCS 4011, pp.303-316,2006
- [15] Noy N.F., McGuinness D.L.: Ontology Development 101: A Guide to Creating Your First Ontology, http://protege.stanford.edu/publications/ontology_development/ontology101.html
- [16] ISO Released version of MOF (ISO/IEC 19502), v.1.4.1: Ch.7.9. MOF Model Constraints, <http://www.omg.org/spec/MOF/ISO/19502/PDF/>
- [17] Bergman M.K. AI3, <http://www.mkbergman.com/913/metamodeling-in-domain-ontologies/>
- [18] Berners-Lee T.: Linked Data, <http://www.w3.org/DesignIssues/LinkedData.html>
- [19] OpenLink Virtuoso Universal Server: Documentation, <http://docs.openlinksw.com/virtuoso/index.html>
- [20] SPARQL Query Language for RDF, <http://www.w3.org/TR/rdf-sparql-query/>
- [21] Rikacovs S., Barzdins J.: Export of Relational Databases to RDF Databases: A Case Study, P.Forbrig, H.Günter(Eds.) Perspectives in Business Informatics Research, Proceedings of 9th International Conference, BIR 2010, Rostock, Germany: LNBIP 64, pp.203–211, 2010.

RDB2OWL: a RDB-to-RDF/OWL Mapping Specification Language

Kārlis ČERĀNS^a and Guntars BŪMANS^{b,1}

^a*Institute of Mathematics and Computer Science, University of Latvia
Raiņa bulvāris 29, Rīga LV-1459, Latvia*

karlis.cerans@mii.lu.lv

^b*University of Latvia, Raiņa bulvāris 19, Rīga LV-1586, Latvia*

guntars.bumans@gmail.com

Abstract. We present a RDB2OWL mapping specification language that is aimed at presenting RDB-to-RDF/OWL mappings possibly involving advanced correspondences between the database and ontology in a human comprehensible way. The RDB2OWL mappings can be regarded as documentation of the database-to-ontology relation. The RDB2OWL language reuses the OWL ontology structure as a backbone for mapping specification by placing the database link information into the annotations for ontology classes and properties. It features reuse of database table key information, user defined and table functions, as well as multiclass conceptualization that is essential for keeping the mapping compact in case of creating a conceptual partitioning of large database tables. We report also on initial implementation experience for a subset of RDB2OWL constructs.

Keywords. Relational databases, RDF, OWL ontologies, mappings

Introduction

With the increasing use of semantic technologies both on world wide web scale, and locally within enterprises, supported by the development of open definitions and standards such as RDF [1], SPARQL 1.1 [2], OWL 2.0 [3] and many others, and the growing number and capacity of the tools for semantic contents management, the need of smooth and efficient information integration between the “old data world” organized primarily along the relational database (RDB) paradigm and the “new information world” inspired by the semantic standards and their support technologies, arises to be of utmost importance. There has been both a long-going and recent intensive research and technology development on bridging these two worlds by RDB-to-RDF/OWL mapping definition languages and techniques, going back to [4] and featuring a number of successful and promising approaches including R₂O [5], D2RQ [6], Virtuoso RDF Views [7], DartGrid [8] as well as recent work on UltraWrap [9] and Triplify [10]. There is W3C RDB2RDF Working Group [11] related to standardization of RDB to RDF mappings, as well as a related published survey of mapping RDBs to RDF [12]. Most of these approaches are concentrating on efficient machine processing of the mappings, often preferably querying RDBs on-the-fly from an SPARQL-enabled

¹ Partially supported by ESF project 2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004

endpoint. Much less attention, however, has been given to creating high-level mapping definitions that are oriented towards readability for a human being and that have a capacity to handle complex database-to-ontology/RDF schema relations.

The concise and human readable RDB-to-RDF/OWL mappings in a situation of an involved schema correspondence is essential e.g. for relational database semantic reengineering task, where a possibly legacy database is to be mapped to RDF/OWL, and the mapping information itself together with the ontology model is required as a documentation of the existing RDB data structure. As an existing approach in this area Semantic SQL [13] can be mentioned, still its relations to the open SPARQL standard, as well as its abilities to handle complex dependencies within a mapping are unclear.

Defining human readable mappings has been long an issue within MOF-centered [14] model transformation community. A model transformation language such as MOF QVT [15], MOLA [16] or AGG [17] (there are many other languages available) may be used for structural presentation of a mapping information; however, these languages are not generally designed to benefit from the mapping specifics that arise in RDB-to-RDF/OWL setting, partially due to simple target model structure (RDF triples). We note an interesting practical experience report of this kind in [18].

The possibility to define RDB-to-RDF/OWL mappings efficiently has emerged as an issue of primary importance also in practical semantic re-engineering of medical domain data in Latvia [19,20,21]. This approach proposes creating ontology (ontologies) for data that are available in a specific domain (e.g., government data, or medical data), using visual graphical notation offered by OWLGrEd [22,23] or UML/OWL profile [24], followed by RDB data integration into the format of the defined conceptual ontology, then followed by providing tools that are able to access the semantic data by means of a visual SPARQL query endpoint [20,25].

In this paper we propose a high level declarative RDB-to-RDF/OWL mapping specification language RDB2OWL that is based on re-using the target ontology structure as a backbone for mapping structure by storing the database link information in the annotations to ontology classes and properties, as well as to the ontology itself. Since the ontology and mapping structures may match imperfectly, some secondary mapping structuring means are made available, as well. The RDB2OWL language features also:

- reuse of RDB table column and key information, whenever that is available,
- concrete human readable syntax for mapping expressions that is very simple and intuitive in the simple cases, and can also handle more advanced cases,
- built-in and user defined functions (including column-valued functions),
- advanced mapping definition primitives, e.g. multiclass conceptualization that avoids the need of specifying long filtering conditions arising due to fixing a missing conceptual structure on large database tables,
- possibility to resort to auxiliary structures defined on SQL level (e.g. user defined permanent and temporary tables, as well as SQL views), still maintaining the principle that the source RDB is to be kept read only.

A previous work by the authors [26,27] on the basis of mapping abstract syntax structures has shown the viability of the central concepts of RDB2OWL by providing an implementation for a subset of RDB2OWL that allowed to efficiently generate the RDF data corresponding to databases from Latvian medical domain [20,21].

In the following sections we introduce three layers of the RDB2OWL mapping language – the Raw language (structure specification, not actually meant for end-users), the Core language (basic mechanisms) and Extended language. Finally, we make some implementation notes and conclude the paper.

The object property maps (*ObjectPropertyMap* instances) establish OWL object property links that correspond to related tables in the database. The tables to be related generally come from source and target class maps of the object property map; they are joined using explicit join condition specification in the object property map's table expression's *filter* attribute, with option to include further linked tables and filters.

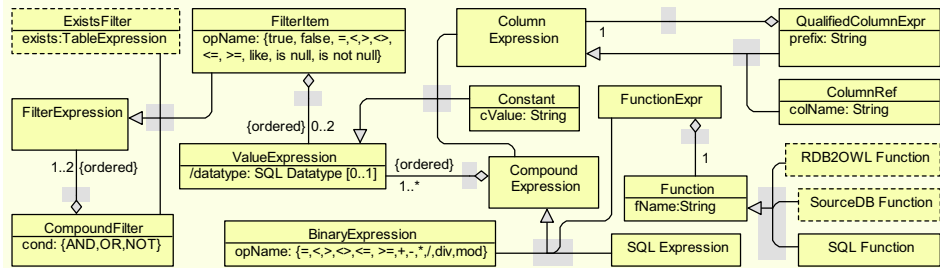


Figure 2. Expression and filter metamodel

1.1. Syntax of Expressions and Entity Maps

Syntactically the RDB2OWL mapping definition is achieved by storing textual class map and property map descriptions in the annotations to the respective OWL classes and properties (we assume a fixed annotation property *DBExpr* is used for this purpose). An OWL class may have several annotations each describing a class map; an OWL datatype or object property may have several annotations describing datatype or object property maps respectively. We start mapping syntax explanation by table expressions.

A table expression description consists of a comma-separated list of reference items, followed by optional filter expression that is separated from the reference item list by a semicolon. Each reference item can be (i) a table name possibly followed by an alias, (ii) a class map reference (one of strings $\langle s \rangle$ or $\langle t \rangle$, optionally preceded by a class map description), or (iii) a table expression enclosed in parentheses possibly followed by an alias string. The filter expression is built in accordance to the abstract syntax of expression and filter metamodel of Figure 2.

The concrete syntax of expressions is based on SQL expression syntax, however not including SQL-style subqueries. Since the RDB2OWL table expressions form a hierarchical structure where every hierarchy level can be identified by an alias, we let the *fully qualified names* (*fqn*, for short) for columns be expressions of the form $a_n.(a_{n-1}.(a_{n-2} \dots (a_1.(a_0.c)))$, where c is a source database table column name, a_0 is a table name and $a_1 \dots a_n$ are prefixes; each prefix is an alias or a class map reference mark. We let a column be identified within a table context not only by its *fqn*, but also by shorter forms (some of prefixes omitted) if that allows unique column identification.

Presenting the syntax we presume the use of parentheses to allow unique identification of abstract syntax structure from the expression text. Some table expressions are:

- *Person*
- *Person P, Address A; P.AdrID_FK=A.AdrID*
- (*Person P, Address A; P.AdrID_FK=A.AdrID*) *PA, (Company, WorksFor; CID=CID_FK); PA.(P.PersonID)= WorksFor.PersonID_FK*
- $\langle s \rangle, \langle t \rangle; \langle s \rangle.AdrID_FK = \langle t \rangle.AdrID$

A class map description is obtained by adding to a table expression description an uri pattern description in the form $\{uri=(\langle item_1 \rangle, \dots, \langle item_k \rangle)\}$, where each $item_i$ is a

value expression (typically, a textual constant, or a reference to a database table column); such pattern describes a conversion to uri form and concatenation of all values *item_i*. Some class map examples are:

- *Person* {uri=('XPerson',PersonID)}
- *Person* *P*, *Address* *A*; *P.AdrID_FK=A.AdrID* {uri=('XPerson',PersonID)}.

An object property map is described by a table expression, containing exactly one source class map reference (a class map reference with *mark*=<*s*>) and exactly one target class map reference (*mark*=<*t*>) within the expression's declaration structure. Some table expressions that describe object property maps are:

- <*s*>, <*t*>; <*s*>.AdrID_FK = <*t*>.AdrID
- (*Person* {uri=('XPerson',PersonID)}) <*s*>, (*Address* {uri=('XAddress',AddressID)}) <*t*>; <*s*>.AdrID_FK = <*t*>.AdrID (<*s*> and <*t*> are class map reference marks preceded by class map definitions).

A class map reference mark <*s*> or <*t*> can be included into object property *p* map expression structure either with a preceding class map description, or without it. The inclusion of a class map description within an object property map expression means defining in-place a new class map that the object property map is going to refer to as its source or target. The most common usage of the construct, however, is without the explicit class map description; in this case the mark <*s*> (resp. <*t*>) refers to the single class map that is ascribed to the domain (resp. range) class of *p*.

A datatype property map is described by a table expression which is required to contain a single <*s*>-marked reference to the source class map, followed by a value expression that is attached to the table expression using a dot notation and further on by an optional datatype specification preceded by the string '^'. Some datatype property description examples are <*s*>.Name, <*s*>.Name^{^xsd:String} and (*XPerson* {uri=('XPerson',PersonID)}) <*s*>.Name.

In the case, if the table expression part for a datatype property map is just <*s*>, we allow omitting it together with the following dot symbol when using a short form of mapping specification. Similarly, the declaration part (together with the following semicolon) may be omitted for an object property map, if it is just <*s*>, <*t*>. These conventions are used in the following example in Section 1.2.

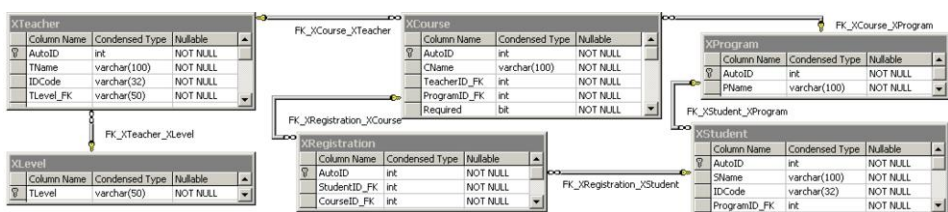


Figure 3. A mini-university database schema

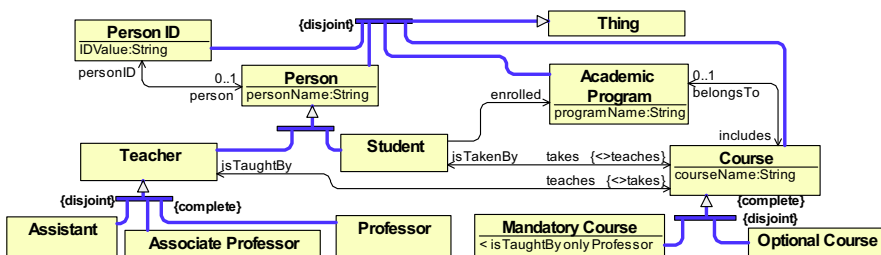


Figure 4. OWL ontology corresponding to the mini-University

- (c) if t consists of items t_1, \dots, t_n with no aliases specified then $C(t) = C(t_1) \cup C(t_2) \cup \dots \cup C(t_n)$; if for some t_i there is an alias specified, the rule (b) is to be applied on this item before (c).

We require that the fully qualified names in the table expression context be distinct; if that is not the case, the table expression is not well formed.

Given a table expression t for a property map m , we denote by $src(t)$ (resp. $trg(t)$) the column prefix within $C(t)$ that ends in $\langle s \rangle$ (resp. $\langle t \rangle$); the requirements on t structure ensure that $src(t)$ (resp. $trg(t)$) is uniquely defined. Note that, for instance, for a table expression $t = (A \langle s \rangle) E1, B \langle t \rangle$ we have $src(t) = E1.\langle s \rangle$ and $trg(t) = \langle t \rangle$.

For a value expression x and a string a we define the a -lifted form of x by replacing every column reference t within the x structure by $a.t$.

The semantics $R(t)$ of a table expression t on the source database \mathcal{S} is defined as a set of rows with columns corresponding to the table expression context $C(t)$ the following way. If there is no filter expression specified as part of t , then

- (a) if t is a table, then $R(t)$ consists of all its rows
- (b) if t is of the form $t' a$ for a table expression t' and an alias a , then $R(t)$ is obtained by renaming $R(t')$ columns via adding the prefix ' a .'
- (c) if t consists of items t_1, \dots, t_n with no aliases involved then $R(t)$ is formed by taking all row combinations from the row sets $R(t_1), \dots, R(t_n)$.

If, however, there is a filter specified as part of t , only the rows that satisfy the filter are retained in $R(t)$, as obtained above.

For every row $r \in R(t)$ there is defined notion of value expression evaluation: the column expressions are looked up within the row; the constants and standard operators have their usual meaning. Let *concat* be a function concatenating all its arguments.

Given the source database schema \mathcal{S} , the corresponding RDF triples are defined for each class map and property map separately.

For a class map or property map m let t be the table expression contained in m and let e be the OWL entity (OWL class or OWL property) that m is ascribed to (we consider only class maps ascribed to OWL classes here). Let r be the *baseURI* specified for the target OWL ontology. In order to form the RDF triples that correspond to \mathcal{S} , we form in each case the row set $R(t)$ by evaluating t on \mathcal{S} . For each row in $R(t)$ we then proceed, as follows:

- if m is a class map, evaluate the expression contained in m 's *uriPattern* attribute obtaining a string value v ; then form the RDF triple $\langle concat(r, v), 'http://www.w3.org/1999/02/22-rdf-syntax-ns\#type', e.entityURI \rangle$;
- if m is an object property map, evaluate the $src(t)$ -lifted form of m 's source class map *uriPattern* to obtain u and the $trg(t)$ -lifted form of m 's target class map *uriPattern* to obtain v , form the triple $\langle concat(r, u), e.entityURI, concat(r, v) \rangle$;
- if m is a datatype property map, let d be the value of m 's *expr* attribute evaluation; let s be obtained by evaluating the $src(t)$ -lifted form of m 's source class map *uriPattern* value. Further on we find dt : XSD datatype corresponding to d the following way:
 - if there is an XSD datatype specified within m , take this datatype
 - if the XSD datatype can be found as a default XSD datatype for d 's SQL datatype, take this datatype
 - if the XSD datatype has been specified as $e.range$, take this datatype
 - if none of the above applies take dt to have $typeName = 'xsd:String'$.

The resulting triple is $\langle concat(r, s), e.entityURI, concat(d, '^^', dt.typeName) \rangle$.

2. The Core Language

We start presenting RDB2OWL core language by augmenting RDB2OWL raw metamodel (Figures 1 and 2) with a possibility to label a reference item (*RefItem* class element) a *top*-constrained element of a table expression. The semantics of such an element *el*, if designated for a table expression *t*, is reflected in the row set $R(t)$ construction for *t* in that only the specified rows (e.g., only a top row) for columns corresponding to *el* is included into $R(t)$ for a combination of values in other *t* columns. Syntactically we write $(XStudent <s>, XProgram <t> \{top\ 1\ PName\ asc\}; PName > ')$ to denote all students and the first program with non-empty name for each of them.

The RDB2OWL core language constructs are summarized in Figure 6. The core language constructs are semantically explained via their translation into the augmented RDB2OWL raw language. The principal novelties here are:

- a more refined table expression structure (each table expression reference list item now may be expressed as list-like navigation item and link structure);
- the primary key and foreign key information within RDB MM; this allows:
 - (i) introducing default entity URI patterns on the basis of RDB schema structure: the default uri pattern for a class map, whose table expression is a single table *X* having a sole primary key column *P*, is defined as $uri('X',P)$; and
 - (ii) avoiding the necessity to specify column names in linked table conditions, if these correspond to a unanimously clear table key information;
- the naming of class maps (*defName* attribute), together with a possibility to refer within a table expression item (a *NamedRef* element) either to such a defined class map, or to the sole un-named class map defined for a certain OWL class *c*; a named reference syntactically is represented as $[[R]]$, where '*R*' is either the name of an owl class, or the defined name of a class map.

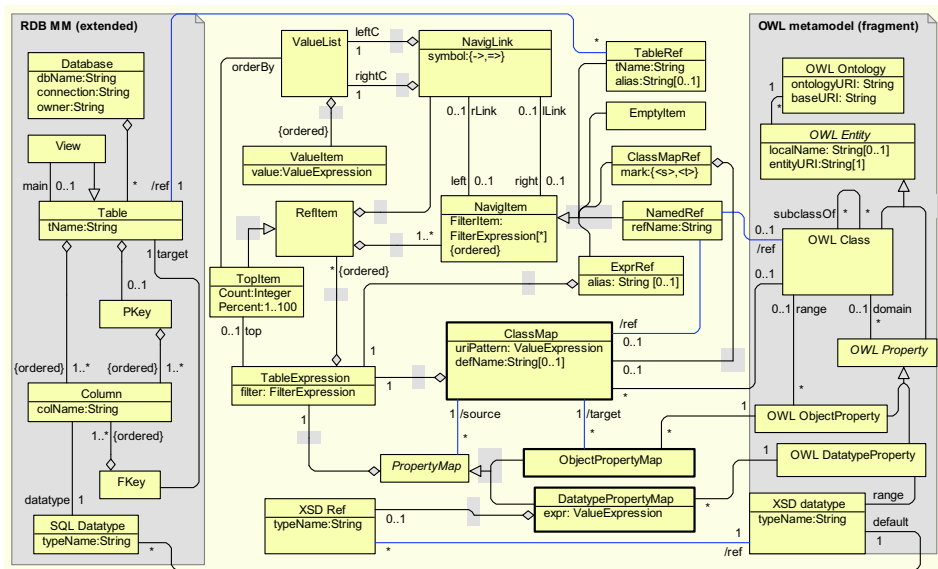


Figure 6. RDB2OWL Core metamodel

The principal building blocks of the RDB2OWL Core metamodel, as in the model Raw version, are class maps and property maps that are based on table expressions,

consisting of reference items. A reference item can contain a singleton list of navigation items thus subsuming the reference item structure of Raw metamodel. The navigation item and navigation link structure of a reference item allows encoding filters in simple table expressions as equality conditions between columns in two adjacent navigation items. For instance, $XStudent[ProgramID_FK]->[AutoID]XProgram$ is a reference item presentation in a navigation item and link form, where $XStudent$ and $XProgram$ are navigation items and $[ProgramID_FK]->[AutoID]$ is a navigation link with $ProgramID_FK$ and $AutoID$ being its left and right value lists (and the sole value items within these lists) respectively. The presented navigation item and link structure can be translated into an equivalent table expression $XStudent\ E1, XProgram\ E2; E1.ProgramID_FK=E2.AutoID$, where $E1$ and $E2$ are unique system generated aliases introduced to avoid potential table name conflicts. The navigation structures can form also longer chains as, for example, the following (*):

$<s>[AutoID]->[StudentID_FK]XRegistration[CourseID_FK]->[AutoID]<t>$.

We allow an empty navigation item only in linked navigation structures (the ones containing link symbol $->$ or $=>$ adjacent to the empty item); in this case the empty item is shorthand for a class map reference $<s>$ or $<t>$, and it is replaced by the reference during the expression's short form unwinding, as explained below in steps 1. ... 4.

For a navigation link $A[F1]->[F2]B$ between two single table items A and B (possibly included in expressions with filters and aliases, or being class map references or named references to single table class maps) the table column list $[F2]$ may be omitted, if it corresponds to the primary key of B . Furthermore, $[F1]$ may be omitted, as well, if there is a single foreign-to-primary key reference in the source RDB from A to B and it is based on the equality of the foreign key columns $F1$ to primary key columns $F2$. For the case when $A[F1]->[F2]B$ is a primary-to-foreign key relation, we allow omitting both $[F1]$ and $[F2]$, in this case presenting the expression as $A=>B$. The above navigation structure (*), given the mini-University database schema of Figure 3, for the property *takes* can then be presented as $<s>=>XRegistration-><t>$.

The table expressions corresponding to the RDB2OWL Core metamodel are transformed into the (augmented) RDB2OWL Raw format via the following steps:

- A. Unwinding of the short form of table expressions (insertion of $<s>$ and $<t>$ class map references, where appropriate) for datatype and object properties, following the rules explained below in steps 1. ... 4.
- B. Insertion of explicit uri pattern definitions for class maps, in place of default uri expressions (including both class maps ascribed to classes and defined inline).
- C. Replacing named references by their referred class maps defined inline.
- D. Insertion of explicit column information definition in navigation links.
- E. Converting the navigation expression structure into reference item structure.

For a table expression reference list structure consisting of expression and navigation items we define its *leftmost* (resp. *rightmost*) item by recursively choosing expression's leftmost (resp. rightmost) reference or navigation item, until an item that is an empty item, a table reference, a class map reference (with or without an explicit class map specification), or a named reference, is found.

The rules for unwinding the short form of a table expression e corresponding to an object property map are, as follows:

1. if e 's reference item list is empty define this list to be $<s>, <t>$;
2. if the leftmost (resp. rightmost) item is an empty item, replace this item by $<s>$ (resp. $<t>$); e.g. any of ' $<s>->$ ', ' $-><t>$ ' and ' $->$ ' is replaced by ' $<s>-><t>$ ' and ' $(XPerson\ <s>)->$ ' is replaced by ' $(XPerson\ <s>)-><t>$ ';

3. The Extended Language

We present here the mapping language and framework extensions that are essential for mapping applicability to practical use cases, maintaining the compact and intuitive mappings. We relate the explanation of these concepts here to their practical use in a case study of using RDB2OWL approach to migration into RDF format of 6 Latvian medical registries [20,21] data, involving 106 source database tables, 1353 table columns (in total more than 3 million rows, altogether 3 GB of data), as reported in [26,27]. The corresponding OWL ontology in the case study had 172 OWL classes, 814 OWL datatype properties and 218 OWL object properties [26].

3.1. Multiclass Conceptualization

The motivation for the feature comes from the cases when some large database table during the mapping is conceptually split into a set of OWL classes, each class responsible for a certain subset of the table columns. For instance, there may be different groups *A*, *B* and *C* of measurements taken during a clinical anamnesis, all recorded into a single table *T* as fields *A1..A10*, *B1..B10* and *C1..C10*. The conceptual model may indicate classes *A*, *B* and *C* corresponding to *T*, with the condition that an ontology individual corresponding to a *T* record has a type *A* (resp. *B*, resp. *C*) if and only if at least one of fields *A1..A10* (resp. *B1..B10*, resp. *C1..C10*) has been filled. This can be solved in RDB2OWL Core notation by adding a filter (*A1 is not null*) OR ... OR (*A10 is not null*) to the class map for *A* and similar filters for *B* and *C*. However, as the field groups may be very large (up to 35 fields in the medicine registries example), writing these conditions becomes a tedious and error-prone task.

To offer an user-friendlier notation for this situation we introduce into RDB2OWL specific decorations that can be attached to the class maps and/or OWL classes. The decoration ‘?Out’ that can be shortened to ‘?’ invokes the semantics of specifying a correspondence to the source database data only for those $\langle x, \text{'rdf:type'}, o \rangle$ instance triples, where a corresponding triple $\langle x, p, y \rangle$ exists for some property *p* whose domain is *o* (i.e. some property that is going out of *o*).

In the considered example, the class maps for each of *A*, *B* and *C* will involve no explicitly specified filtering condition, but they will be decorated by ‘?’.

There are also ‘?In’ and ‘?Any’ decorations available for class maps and OWL classes, requiring existence of incoming or any property for an OWL class instance. Furthermore, we provide ‘?Dom’ decorations for property maps and ‘?Ran’ decorations for object property maps that are used to validate that the subject or object of a triple corresponding to the property (or a property map) has its ‘rdf:type’- triple to the corresponding property domain resp. range ontology class.

For the Latvian medicine registries migration the multiclass conceptualization feature has been extensively used: the decoration ‘?Out’ has been attached to 54 out of 172 OWL classes having 542 outgoing datatype properties (out of the total 814 OWL datatype properties in the case study).

3.2. Auxiliary Database Objects

As can be observed from the presented examples, the mapping language provides a format of mapping expression that is more convenient than a mapping encoding within

SQL; however, for the sake of universality, the possibility to create additional tables and the full power of SQL needs to be made available within mapping definition.

The practical need in auxiliary database objects has been observed for *Numbers* or *Tally* table (cf. [30]) as well as the classifier tables that have not been explicitly created during the relational database design time. We propose creating a new auxiliary database schema for the mapping definition, where the auxiliary tables and views, as well as some temporary data structures, if they were necessary, could be stored. The mapping annotations within the ontology then contain the information pointing to:

- 1) the auxiliary database,
- 2) the information about the code that is to be executed every time the RDF triple generation from the source database is run (this may be e.g. some complicated data processing, or filling temporal tables).

3.3. Built-in and User-defined Functions

There is a number of built-in RDB2OWL functions that allow a higher level expression forming in the mapping specification. We mention here the single-argument conversion functions: `#varchar(_)`, `#xvarchar(_)` (converts to varchar and eliminates leading and trailing spaces), `#uri(_)`, as well as multi-argument `#concat(...)`, `#xconcat(...)` and `#uriConcat(...)` that combine conversion and concatenation. Furthermore, `#exists(...)` takes any number of arguments and returns 1 iff at least one of them is not null and `#if(a,b,c)` chooses the value of *b* or *c* depending on *a* value being 1 or 0. The list of built-in functions is to be finalized within upcoming RDB2OWL reference manual.

An important RDB2OWL feature is a possibility to define user-defined functions. Technically these functions are placed in *DBExpr*-annotations on the ontology level. A function may take a fixed number of arguments, whose names start with '@' symbol. A simple function body consists of a value expression (including optional data type specification), as, for instance `BoolT(@X) = (#if(@X, 'true', 'false') ^xsd:Boolean)` that converts a bit value 0 resp. 1 into `'true'^xsd:Boolean` resp. `'false'^xsd:Boolean`.

A principal feature of user-defined functions is a possibility to include also a table expression within the function body (the function body then corresponds to datatype property syntax). If a function $f(@X) = (F;filter.val)$ is called as $f(V)$ in the context of a table *A*, we evaluate the expression $(A,F;filter[V/@X]).val[V/@X]$, where $[V/@X]$ denotes the substitution of the value *V* for the variable *@X*. As a result, a single or several resulting values are obtained; all of the resulting values are considered for making RDF triples corresponding to the source database information.

A practically important example of this kind has been the function `split4(@X) = ((Numbers;len(@X)>=N*4).substring(@X,N*4-3,4))`, where *Numbers* is a table with single integer column filled by numbers from 1 to 8000 [30]. The application `split4(FieldX)` of such a function ensures splitting of a character string into a set of its blocks of length 4 (e.g., `'199219982004' → {'1992', '1998', '2004'}`). Note that the *Numbers* table allows implementing a wide range of string splitting tasks [30].

A simpler application of table expressions in functions (not creating “column-valued” functions) would correspond to “translation table” functionality of D2RQ [6].

The full version of RDB2OWL admits also referring to database tables themselves as “translation functions”, as well as built-in and user-defined aggregate functions not detailed here. We note that the aggregate function functionality (although, using lower level constructs) can be achieved also using SQL means, as described in Section 3.2.

4. Implementation and Experience

The RDF triple generation from RDB schema has been implemented for a subset of RDB2OWL language constructs, on the basis of manually created abstract syntax representation of mapping information. The implemented language constructs involve raw RDB2OWL model (with a restriction of a single table for a class map), with some support for automated URI pattern construction, as well as multiclass conceptualization and reliance on external database schema. The initial modeling and implementation effort using RDB2OWL approach has been reported in [26,27], where we report also on successful experience of semantic re-engineering of Latvian Medical registries data [20,21]. The mapping of the registries has been implemented by generating SQL statements that generate the corresponding RDF triples from the source RDB; the running time for approximately 42.8 million RDF triple generation on Intel Mobile Core 2 Duo T6570 processor running Windows Vista, 3 GB of RAM has been less than 19 minutes [27] – a result that is completely satisfactory for the foreseen application.

There is work in progress on creating a context-aware parser for the concrete syntax of RDB2OWL mapping language, as well as for supporting the full set of language features presented here.

Our primary interest is in running the RDF triple generation in a batch mode (this is sufficient for a wide number of applications, e.g. in government data area). However, the RDB2OWL mapping information can enable alternative implementations, possibly with generating the queries over the source SQL database online within the context of SPARQL queries over the target RDF data store (this should be at least theoretically possible for a large subset of RDB2OWL constructs; the temporal tables and user-defined aggregate functions are going to be among the first exclusions).

5. Conclusions

We have presented RDB2OWL approach to RDB-to-RDF/OWL mapping specification that re-uses the ontology structure as the backbone of the mapping specification by putting the mapping definitions in the OWL ontology entity annotations.

As the mapping examples show, the approach can be used for a convenient mapping definition. Combining the power of the RDB2OWL mapping definition approach with visual ontology modeling means such as OWLGrEd [22,23] notation and editor can be a viable mechanism for the RDB semantic re-engineering task. Since the ontology annotation mechanism is a part of ontology definition means, the RDB2OWL-annotated ontologies can be used also outside the concrete ontology editor.

The RDB2OWL approach has been successfully used for a “real-size” task of semantic re-engineering of databases in Latvian medical domain. There is work in progress towards the implementation of the full set of RDB2OWL constructs, including RDB2OWL parsing on a concrete syntax level and integrating into OWLGrEd editor.

It seems to be a plausible and interesting task to adapt the mapping constructions considered here also for RDF/OWL-to-RDF/OWL mapping definition that may be useful in transformation from the “technical data ontology” to the conceptual one, after the initial data – be these in RDB or some other format – have been exposed to the RDF format using a straightforward and technical structure preserving embedding.

References

- [1] Resource Description Framework (RDF), <http://www.w3.org/RDF/>
- [2] SPARQL 1.1 Query Language, <http://www.w3.org/TR/2010/WD-sparql11-query-20100601/>
- [3] OWL 2 Web Ontology Language, Structural Specification and Functional-Style Syntax <http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/>
- [4] T.Berners-Lee, Relational Databases on the Semantic Web, <http://www.w3.org/DesignIssues/RDB-RDF.html>, 1998.
- [5] J.Barrasa, O.Corcho, G.Shen, A.Gomez-Perez: R2O: An extensible and semantically based database-to-ontology mapping language. In: *SWDB'04, 2nd Workshop on Semantic Web and Databases* (2004).
- [6] D2RQ Platform, <http://www4.wiwiw.fu-berlin.de/bizer/D2RQ/spec/>
- [7] C.Blakeley: RDF Views of SQL Data (Declarative SQL Schema to RDF Mapping), OpenLink Software, 2007.
- [8] W.Hu, Y.Qu: Discovering Simple Mappings Between Relational Database Schemas and Ontologies, In *Proceedings of 6th International Semantic Web Conference (ISWC 2007), 2nd Asian Semantic Web Conference (ASWC 2007)*, LNCS 4825, pages 225-238, Busan, Korea, 11-15 November 2007.
- [9] Sequeda, J.F., Cunningham, C., Depena, R., Miranker, D.P. Ultrawrap: Using SQL Views for RDB2RDF. In *Poster Proceedings of the 8th International Semantic Web Conference (ISWC2009)*, Chantilly, VA, USA. (2009)
- [10] Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., AumueLLer, D.: Triplify: Light-weight linked data publication from relational databases. In *Proceedings of the 18th International Conference on World Wide Web* (2009).
- [11] W3C RDB2RDF Working Group, <http://www.w3.org/2001/sw/rdb2rdf/>
- [12] A Survey of Current Approaches for Mapping of Relational Databases to RDF, http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf
- [13] Semantic SQL: <http://semanticsql.com/>
- [14] OMG's MetaObject Facility, <http://www.omg.org/mof/>
- [15] MOF QVT, <http://www.omg.org/spec/QVT/1.0/>
- [16] MOLA project, <http://mola.mii.lu.lv/>
- [17] The <AGG> Homepage, <http://user.cs.tu-berlin.de/~gragra/agg/>
- [18] S.Rikacovs, J.Barzdins, Export of Relational Databases to RDF Databases: a Case Study, in P. Forbrig and H. Günther (eds.), *Perspectives in Business Informatics Research*, Springer LNBIP **64** (2010), 203-211.
- [19] J.Barzdins, G.Barzdins, R.Balodis, K.Cerans, et.al., Towards Semantic Latvia. In *Communications of 7th International Baltic Conference on Databases and Information Systems* (2006), 203-218.
- [20] G.Barzdins, E.Liepins, M.Veilande, M.Zviedris: Semantic Latvia Approach in the Medical Domain. *Proc. 8th International Baltic Conference on Databases and Information Systems*. H.M.Haav, A.Kalja (eds.) Tallinn University of Technology Press (2008), 89-102.
- [21] G.Barzdins, S.Rikacovs, M.Veilande, and M.Zviedris, Ontological Re-engineering of Medical Databases, *Proc. of the Latvian Academy of Sciences*, Section B, Vol. **63** (2009), No. 4/5 (663/664), 20-30.
- [22] J.Barzdins, G.Barzdins, K.Cerans, R.Liepins, A.Sprogis: UML Style Graphical Notation and Editor for OWL 2, in P. Forbrig and H. Günther (eds.), *Perspectives in Business Informatics Research*, Springer LNBIP **64** (2010), 102-113.
- [23] OWLGrEd, <http://owlgred.lumii.lv/>
- [24] Ontology Definition Metamodel. OMG Adopted Specification. Document Number: ptc/2007-09-09, November 2007. <http://www.omg.org/docs/ptc/07-09-09.pdf>
- [25] G.Barzdins, S.Rikacovs, M.Zviedris: Graphical Query Language as SPARQL Frontend. In Grundspenkis, J., Kirikova, M., Manolopoulos, Y., Morzy, T., Novickis, L., Vossen, G. (Eds.), *Proc. of 13th East-European Conference (ADBIS 2009)*, Riga Technical University, Riga (2009), 93-107.
- [26] G.Būmans, K.Čerāns, RDB2OWL: Mapping Relational Databases into OWL Ontologies - a Practical Approach. *Databases and information systems: Proc. of the Ninth International Baltic Conference, Baltic DB&IS 2010*, Riga, Latvia, July 5-7, 2010. Riga, University of Latvia (2010), 393-408.
- [27] G.Būmans, K.Čerāns, RDB2OWL: a Practical Approach for Transforming RDB Data into RDF/OWL, in *Proceedings of the 6th International Conference on Semantic Systems*, Graz, Austria, September 2010, ACM International Conference Proceeding Series, ISBN 9781450300148 (2010) Article No.25.
- [28] G.Barzdins, J.Barzdins, K.Cerans: From Databases to Ontologies, *Semantic Web Engineering in the Knowledge Society*; J.Cardoso, M.Lytras (Eds.), IGI Global, (2008), 242-266.
- [29] G.Bumans, Mapping between Relational Databases and OWL Ontologies: an Example, *Scientific Papers, University of Latvia*, Vol. **756**, Computer Science and Information Technologies (2010) 99-117.
- [30] J.Moden. The "Numbers" or "Tally" Table: What it is and how it replaces a loop. <http://www.sqlservercentral.com/articles/T-SQL/62867/>

Spatial Ontology in Factored Statistical Machine Translation

Raivis SKADINŠ

Tilde, Latvia

Abstract. This paper presents a statistical phrase-based machine translation system which is enriched with semantic data coming from a spatial ontology. Paper presents the spatial ontology, how it is integrated in statistical machine translation system using factored models and how it is being evaluated using both automatic and human evaluation. Spatial information is added as a factor in both translation and language models. SOLIM spatial ontology language is used to implement ontology and to infer necessary knowledge for training statistical machine translation system. The machine translation system is based on Moses toolkit.

Keywords: Spatial Ontology, Statistical Machine Translation, RCC8.

Introduction

There are several different state-of-the-art approaches for building Machine Translation (MT) systems. Most popular methods used are knowledge based methods and corpus based methods. First MT systems where knowledge or rule based MT systems. They use human created rules, dictionaries and knowledge bases to describe grammars of languages and to specify translation rules. Corpus based MT systems do not need human created rules, grammars and dictionaries; they use statistical methods to analyze large amounts of human translated texts. Statistical MT (SMT) systems extract statistical data about translations and use these data to translate new texts.

Both knowledge and corpus based methods have their advantages and disadvantages. Knowledge based MT systems perform well if they have all necessary knowledge. Natural languages are very complex with many exceptions, uncertainties and ambiguities in rules and dictionaries. Therefore it is not possible to provide all necessary knowledge for knowledge based MT systems and these systems typically are working with incomplete data and that leads to many mistakes.

Corpus based MT systems automatically extract statistics from huge amounts of translated texts (parallel corpus). Just more and more training data are needed to improve corpus based MT systems. The main weakness of corpus based MT methods is the availability of reasonably sized parallel corpus. Pure SMT methods [1] [2] are not using any semantic or linguistic information including information about morphology and word inflections.

The latest SMT methods are using different additional knowledge to build more sophisticated statistical models thus improving MT quality. Typically morphological

and syntactical annotations are added to training data, and SMT systems learn translation and language models which benefit from this annotation. Typical examples of such SMT methods are factored phrase-based SMT [3], tree-based SMT [4] [5] [6], treelet SMT [7].

This paper presents ongoing research which is aiming to add some semantic knowledge to SMT. MT systems could benefit from various kinds of semantic knowledge in various stages of translation or training processes. Semantic information might be used in word breaking, part-of-speech tagging, syntactic disambiguation, word sense disambiguation etc. All mentioned areas are clearly distinguished in rule based MT systems, but they are somewhat vague in SMT. There is no words sense disambiguation component in SMT, but SMT models are built so that they can deal with word ambiguities. Although various kinds of semantic knowledge could be used to improve various translation aspects, such as word sense disambiguation, translation selection or phrase reordering, this paper is focusing on using of spatial information for word sense disambiguation in factored phrase-based SMT. Spatial ontology language developed in SOLIM¹ project is used to implement spatial.

In addition to this introductory chapter, this paper is divided into four main chapters. In chapter one, a brief introduction to SOLIM ontology language and implementation of the ontology is presented. Chapter two presents integration of the spatial ontology in SMT system. Chapter three gives information about both automatic and human MT evaluation methods used to evaluate impact of spatial knowledge on quality of MT system. In chapter four current results are presented and further research is outlined.

1. SOLIM Language and Ontology Implementation

This chapter is based on SOLIM project deliverable [8] describing ontology language design and it gives an overview of ontology used in SMT system described later.

Spatial information, as well as information in general, is mostly implicit in nature. Its character is descriptive and meaningless without its most skilled interpreter: the human brain. If we want to use spatial information to train better SMT models, then this information should be described in a machine-readable way that enables artificial agents to at least act as if they would have some understanding of the information being processed. The Web Ontology Language (OWL), a W3C standard, is a state-of-the-art means to achieve this.

OWL provides the means to go beyond keywords and specify the meaning of the resources described. However, not all concepts are equally easy to define. One of the concepts that are limited by OWL's structure is space. Spatial information is important for word sense disambiguation in MT. The problem is not so much that OWL is not capable of handling spatial representations, but rather that it is not designed for this type of encoding or reasoning and this makes it rather difficult to express some spatial properties and relationships and to be able to reason with these concepts.

¹ SOLIM project. EUREKA's Eurostars Programme. no.: E! 4365. <http://www.solim.eu>

Since the worlds described in OWL are a reflection of the real-world around us, concrete concepts in OWL implicitly have a spatial property. Yet, the OWL language does not allow expressing the spatiality of those properties in a useful way. This is the goal of the SOLIM project: to expand the OWL language to allow for meaningful explicit expression of spatial properties.

This paper describes the set of essential features that the SOLIM language presents. There are two types of aspects regarding spatiality - the absolute location in space, in the form of coordinates, geo references etc. and the relative spatiality, i.e. entities of which the location is only defined as binary properties between concepts or their instances. As geographical coordinates are not typical for texts which are translated using MT systems, relative aspect of spatiality is more important for MT systems.

When reasoning with and about spatial properties, the question is raised what the notions of space or spatiality mean. Even though an exhaustive answer to this question is far beyond the purpose of this paper, a practical definition of spatiality is needed.

A first aspect of space relevant for the SOLIM language is that it relates to the representation of both absolute and relative locations and a second aspect is that it can represent dimensions (size, shape, etc). Such basic definitions of space in OWL would enable it to represent the constraints and properties of objects and is not explicitly supported in the version 1.0 of OWL-DL. The SOLIM project defines a language that supports the definition of both absolute and relative spatial properties, and – as a result – enables the representation of Region Connection Calculus (RCC) relations.

There are many types of spatial relationships, and an often used set for representing spatial relationships is the RCC-8 calculus. However, RCC-8 is not simple to implement in ontology languages because it requires certain adaptations of them. The goal of SOLIM project is to extend the language of choice; OWL, so that it can represent RCC-8 relationships properly. This involves an adaptation of the OWL language and the reasoner.

In considering possibilities and requirements for a comprehensive ontological scheme for spatial representation, it is essential to incorporate the very extensive work that has been carried out primarily within the tradition of qualitative spatial representation. As a starting point for discussion we take two independently developed views of spatial relationships: the Region Connection Calculus, RCC, proposed by Randell, Cui & Cohn [9] and the set of topological constraints proposed by Egenhofer [10]. These involve stating basic spatial relationships that may hold between spatial entities and working out ways of both reasoning with them and applying them to complex spatial configurations. Although Egenhofer describes his relations in purely topological terms and draws on set theory (regions as sets of points) for definitions, while Randell et al. draw on a topology of regions with spatial parts and start from the connection relation alone, there are clear similarities between them. Their ontological commitments are, however, somewhat different concerning the particular kinds of spatial objects assumed. The relations proposed are set out in Figure. 1. with the names employed in both approaches.

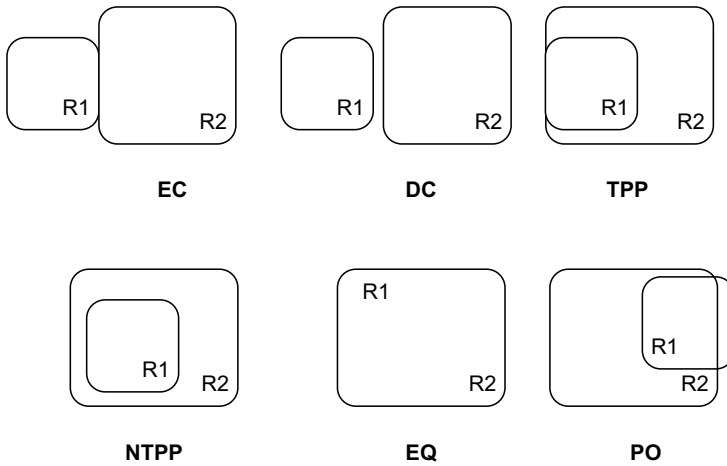


Figure 1. The standard ‘base relations’ of RCC and similar calculi: disjoint/disconnected (DC), meet/external connection (EC), overlap/partial overlap (PO), equal (EQ), covered by/tangential proper part (TPP), inside/non-tangential proper part (NTPP). The inverses of the latter two are not shown: covers/tangential proper part inverse (TPP⁻¹), contains/non-tangential proper part inverse (NTPP⁻¹).

When formalizing them, both approaches make central use of the topological relationship of connection and typically begin their accounts with an enumeration of the distinct ways in which spatial entities can be related spatially: thus leading to these standard eight relations. There are, however, a number of ways of formalizing this view of spatial entities and their relations. While the mereotopological approach common in ontology begins by axiomatizing the relations and their properties in a first-order logical language (in terms of connection and parthood), the computational behavior of such formalization is not good because of its expressiveness.

The full first-order theory of RCC inherits undecidability and so various, more constrained, adaptations of the full theory have been defined. Within qualitative spatial reasoning, therefore, formalizations of spatial relationships that draw on the formal properties of other mathematical accounts is a very active area. By this means, one attempts to achieve more attractive computational properties that are more amenable to practical reasoning tasks. This also appears motivated cognitively in that certain tasks are readily performable by humans and others less so; such evidence can further constrain the kinds of modeling decisions that may eventually be made in formulating a realistic ontology of space.

As was described by Katz [11], RCC8 logic can be expressed in OWL DL with the exception of reflexive roles. Reflexive roles are required since the main property required to express RCC relationships – the “connects” property – has to be reflexive as any region is connected to itself. With the emergence of OWL 2.0, reflexive roles are available and therefore OWL2 is sufficiently expressive to allow the formal description of RCC relations.

To express RCC8 relationships the following translations (see formulas 1 to 6) are made:

$$EC(X, Y) \rightarrow \begin{cases} \forall R. X \subseteq \exists R. \neg Y \\ Z_5 \equiv X \cap Y \end{cases} \quad (1)$$

The EC relationship is slightly more complex than for example DC (see formula 2) or EQ (see formula 5) as it involves the creation of two named classes (R and Z_5). These classes are defined as follows:

1. The class consisting of the points that only connect to X (the interior of X)
2. The class consisting of the points that have at least one connection to the region outside Y (the complement of the interior of Y).

Furthermore, the first of these two classes is expressed to subsume the second. Thus, to express external connectedness, the region outside Y has to contain the interior of X.

$$DC(X, Y) \rightarrow X \subseteq \neg Y \quad (2)$$

$$TTP(X, Y) \rightarrow \begin{cases} X \subseteq Y \\ Z_1 \equiv X \cap \exists R. \neg Y \end{cases} \quad (3)$$

$$NTTP(X, Y) \rightarrow X \subseteq \forall R. Y \quad (4)$$

$$EQ(X, Y) \rightarrow X \equiv Y \quad (5)$$

$$PO(X, Y) \rightarrow \begin{cases} Z_2 \equiv \forall R. X \cap \forall R. Y \\ Z_3 \equiv X \cap \neg Y \\ Z_4 \equiv \neg X \cap Y \end{cases} \quad (6)$$

Ontology implemented in SOLIM language allows representing spatial information about different object and inferring implicit spatial information. Currently SOLIM project partners are evaluating suitability of SOLIM language for machine translation and for image search. The current implementation of ontology for machine translation contains spatial information about all continents, countries and major cities in English, Latvian and Lithuanian. Reasoner can use the ontology to infer spatial relations between objects.

2. Integration of Spatial Ontology into SMT

The proposed MT system is based on statistical machine translation, where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual and monolingual text corpora. The process of statistical model generation is called training, and typically involves analyzing vast amounts of parallel sentence pairs in two languages (bilingual corpus) in order to

generate a translation model, as well as analyzing the target language (monolingual corpus) in order to generate a language model.

The translation model represents a certain statistical model of how a source language (L1) is translated into a target one (L2). The translation model can operate in the domain of words, phrases and more complex structures (for example, syntax trees), and in general is responsible for the adequacy of translation. The language model represents the knowledge about the target language, its sentence and phrase structure and is in general responsible for the fluency of translation.

The current state-of-the-art approach to SMT, so-called phrase-based models, are limited to the mapping of small text chunks (phrases) without any explicit use of linguistic information, may it be morphological, syntactic, or semantic. Such additional information has been demonstrated to be valuable by integrating it in pre-processing or post-processing. However, a tighter integration of linguistic information into the translation model is desirable for two reasons:

1. Translation models that operate on more general representations, such as lemmas² instead of surface forms of lexical units, can draw on richer statistics and overcome the data sparseness problems caused by limited training data.

2. Many aspects of translation can be best explained on a morphological, syntactic, or semantic level. Having such information available to the translation model allows the direct modeling of these aspects. For instance: reordering at the sentence level is mostly driven by general syntactic principles, local agreement constraints show up in morphology, etc.

Moses [12] is a statistical machine translation system that allows automatic training of translation models for any language pair. All you need is a collection of translated texts (parallel corpus). Moses has the following distinct features:

- Beam-search: an efficient search algorithm finds quickly the highest probability translation among the exponential number of choices;
- Phrase-based: the state-of-the-art in SMT allows the translation of short text chunks;
- Factored: lexical units may have factored representation (surface forms, lemma, part-of-speech, morphology, lexeme classes...).

Moses framework is an extension of the phrase-based approach. It adds additional annotation at the lexical unit level. A lexical unit in our framework is not anymore only a token, but a vector of factors that represent different levels of annotation. The training data (a parallel corpus) has to be annotated with additional factors. For instance, if it is necessary to add part-of-speech information on the input and output side then part-of-speech tagged training data are required. Typically this involves running automatic tools on the corpus, since manually annotated corpora are rare and expensive to produce.

The proposed SMT system is based on the Moses SMT system, which features factored translation models that allow integrating additional layers of data tightly into the process of translation. The translation process translates the source (input) text into the target (output/translation) using the trained language and phrase (translation) models (see Figure. 2.). The system output is then compared with a reference (human)

² Lemma is the canonical form, dictionary form, or citation form of a particular word (surface form), for example 'read' is a lemma of surface form 'reading'.

translation to evaluate the translation quality using automated metrics, such as BLEU score [13].

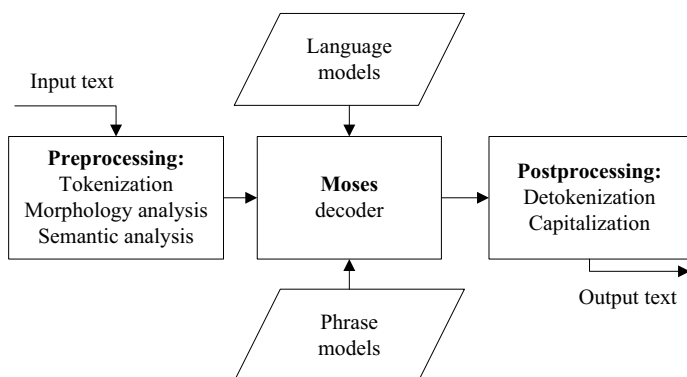


Figure 2. Translation process.

The training process (see Figure. 3.) involves processing a parallel training corpus to generate the phrase model and a monolingual training corpus to generate the language model.

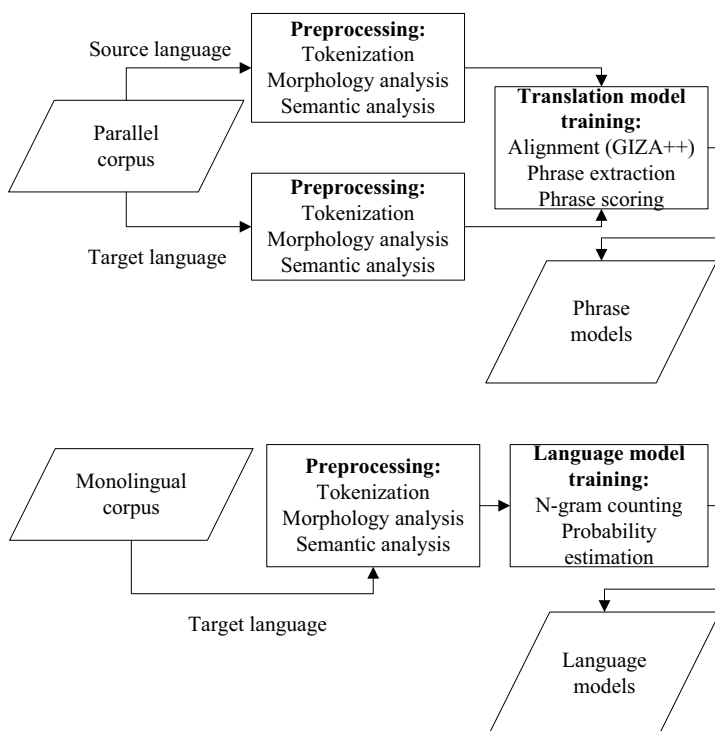


Figure 3. Training process.

Text pre-processing is used both for training and translation tasks. During training, both parallel and monolingual training corpora are preprocessed. During translation, the input text is pre-processed in the same way as for training.

Pre-processing includes the following steps:

- Sentence splitting;
- Tokenization (splitting of text into the smallest translation entities, e.g., words, multiword entities, punctuation symbols);
- Case processing (e.g., lowercasing the first word of sentence);
- Morphologic analysis (optionally);
- Semantic analysis (using the semantic reasoner to augment input tokens with semantic factors).

The inclusion of morphological and semantic analysis depends on the characteristics of the source and target languages and of the translation task itself. Separate pre-processor must be implemented for the source and target languages.

The post-processing works with the output of Moses decoder (translation target language) to convert it to orthographically correct text. It involves the following tasks:

- De-tokenization (gluing of tokens to comply with orthography, e.g., no whitespace before comma);
- Case processing (e.g., capitalizing the first word of each sentence);
- Sentence gluing.

Semantic data has been integrated into the SMT system as additional factors on both source and target language side. By factoring semantic information into the source language, the translation accuracy is improved by resolving semantic ambiguities in the source language. Similarly, by factoring semantic information into the target language, the translation fluency is improved by resolving semantic ambiguities in the target language. The use of factors implies a certain tag set for each factor. The input text will be analyzed and tagged with the information from a knowledge base inferred by the semantic reasoner. Semantic factors, or tags, will be produced for each relevant token in the input text. One or several tags can be used, depending on the particular ambiguities that are targeted. For example, an English place name *Georgia* is ambiguous. It might refer to the US state or the country in Caucasian region. Consider the following ambiguous sentences in English:

1. *The president of Georgia visited Lithuania yesterday.*
2. *There are Lithuanians living in Georgia and other states.*
3. *A recent court ruling gives Georgia, Alabama and Florida three years to resolve their conflict over use of the water from Lake Lanier.*
4. *Experts have failed to travel to Georgia at the Tbilisi international airport.*
5. *Azerbaijan has bad relations with Armenia, so it cannot afford to have such relations with Georgia.*
6. *US president visited Georgia last month.*

It is quite obvious that in examples 1, 4 and 5 *Georgia* stands for the country, but in examples 2 and 3 it stands for the US state. Example 6 is very ambiguous and it is not possible to determine what is meant by *Georgia* without a wider context.

Using typical SMT training methods without tags to train translation models on big amount of parallel sentences results in *Georgia* translated by several different Latvian translation equivalents with different probabilities. The results are as in Table 1.

So, there are 2 translations with a quite high probability, these both are valid translations. The first is a translation of the country *Georgia*, the second is a

translation of the state *Georgia*. Other translations are with very low probability and come from errors in training data.

Table 1. Simple translation probabilities of word *Georgia*.

English	Latvian	probability
Georgia	Gruzija	0.45
Georgia	Džordžija	0.42
Georgia	Grūzija	0.08
Georgia	...	
Georgia	Tibilisi	0.001

If probabilities are calculated for phrase translations some more hints could be obtained. See Table 2.

Table 2. Simple translation probabilities of phrases containing word *Georgia*.

English	Latvian	probability
President of Georgia	Gruzijas presidents	0.9
Georgia Bulldogs	Džordžijas Buldogi	0.8
	...	
President of Georgia	Džordžijas presidents	0.000003

This means that phrase *president of Georgia* will be almost sure translated as *Gruzijas prezidents* and not *Džordžijas prezidents* and *Georgia Bulldogs* will be surely translated as *Džordžijas Buldogi* not *Gruzijas Buldogi*. It can be seen that neighboring lexical units in the context can help to disambiguate phrase translation. Tags can be also assigned to ambiguous lexemes to be disambiguated. Typically part of speech tags are used in factored SMT. For example, if translation probabilities of an English lexeme ‘catch’ are calculated, the following results in Table 3 might be obtained.

Table 3. Simple translation probabilities of words.

English	Latvian	probability
catch	ķert	0.8
catch	loms	0.1
	...	
catch	aizbīdnis	0.01

But if lexemes are tagged with part of speech information then the following results in Table 4 might be obtained.

Table 4. Factored translation probabilities of words.

English	Latvian	probability
catch N	ķēriens	0.4
catch N	loms	0.3
catch V	zvejot	0.5
catch V	ķert	0.4
	...	
catch N	aizbīdnis	0.01

The second table is more precise and more useful in translation, because it gives context dependent and more reliable probabilities.

Other types of tags can be also used to improve an SMT system. There are several ways how lexical units can be tagged with the information obtained from the spatial ontology. For example, if there is a sentence containing lexical units *X* and *Y*

and the information about a relation Z between X and Y can be obtained from the knowledge base, then the lexical unit X can be tagged with tag $X(Z.Y)$. For example, if there is a sentence:

Azerbaijan has bad relations with Armenia so it cannot afford to have such relations with Georgia

a lexeme *Georgia* can be tagged *EC.Armenia* because it can be derived from the ontology that there is a relation *EC* between these two lexemes *Georgia* and *Armenia*. If there is a sentence:

Experts have failed to travel to Georgia at the Tbilisi international airport

the lexeme *Georgia* can be tagged *NTPPi.Tbilisi* because it can be derived from the ontology that there is a relation *NTPPi* between the two lexemes *Georgia* and *Tbilisi*.

SMT training data annotated with such tags will lead to better translation probabilities which are context dependent. The following results could be obtained:

Table 5. Factored translation probabilities of word *Georgia*.

English	Latvian	probability
Georgia EC.Armenia	Gruzija	0.7
Georgia NTPPi.Tbilisi	Gruzija	0.8
Georgia TPP.Caucasus	Grūzija	0.08
Georgia EC.Florida	Džordžija	0.8
Georgia NTPP.US	Džordžija	0.5
Georgia NTPP.US	Gruzija	0.4

The first line in the Table 5 means that a probability of a translation equivalent for the lexeme *Georgia* is *Gruzija* is 0.7, given that there is also a lexeme *Armenia* in the same sentence and there is *EC* relation between the two lexemes *Georgia* and *Armenian* in the ontology.

The described approach is still probabilistic and not knowledge-based. As it can be seen from the Table 5, the fact that there is an *NTPP* relation between the two lexemes *Georgia* and *US* which are in the same sentence does not mean that *Georgia* must be translated as *Džordžija*. *US* might be equally often used together with state *Georgia* and country *Georgia*.

Annotation must be performed both for training corpus and for translation input. During the translation process the factors provide additional information to the decoder, which helps to choose the appropriate translation equivalent. For annotating the data it is necessary to derive the information – the relevant tags, from the spatial ontology.

For this, firstly, the ontology must be able to be queried for all the ontological entities – a list of them: e.g. *GetAllSpatialObjects*: a query to the spatial ontology. This method returns a list of names.

Secondly, after all the relevant lexical units are identified and marked in the texts, it is necessary to derive further information from the spatial ontology (relations between identified lexical units), e.g. *GetSpatialRelations(A, B)*: a query to the spatial ontology to get knowledge from it. This method have 2 parameters (names of spatial objects from spatial ontology) and returns a list of relations inferred by the spatial reasoned (it does not state anything about the nature of relations that are false or are unknown). For example, it is important to know, that *Tbilisi* is a city and it is a city in *Georgia* (*Georgia NTPPi Tbilisi*). This information can influence translation results

and will be used in the training process. Another query *GetSpatialRelations(Georgia,Armenia)* returns the relation *EC* only if there is enough information in the ontology to infer this relation (using open world assumption), whereas the query *GetSpatialRelations(Georgia,Latvia)* returns the relation *DC* if this relation can be inferred (using open world assumption) and *GetSpatialRelations(Georgia,Latvia)* returns nothing if there is no enough information in the ontology to infer *DC* relation.

Implemented methods for ontology querying are used to annotate both SMT training data and MT input text with factors containing spatial knowledge. For all of trained SMT systems the parallel training corpus includes publicly available DGT-TM³ and OPUS corpora [13], as well as proprietary localization corpus obtained from translation memories that were created during localization of software content, appliance user manuals and software help content. Word and phrase translations from bilingual dictionaries were additionally included to increase word coverage. Total size of English-Latvian parallel data is 3.23 M sentence pairs. Monolingual corpora were prepared from the corresponding monolingual part of parallel corpora, as well as news articles from Web for Latvian and European Parliament Proceedings and News Commentary⁴ for English. Total size of Latvian monolingual corpus is 319 M words and 521 M words for English. The evaluation and development corpora were prepared separately. The same mixture of different domains and topics representing the expected translation needs of a typical user for both corpora. The development corpus contains 1000 sentences, while the evaluation set is 500 sentences long.

3. Evaluation

Currently we have SMT system which is enriched with semantic data coming from a spatial ontology. Spatial knowledge definitely gives an improvement of SMT system. But this improvement has to be formally evaluated using MT quality evaluation metrics, such as BLEU score [14] or HTER [15].

The BLEU scores for both baseline SMT system and SMT system with a spatial knowledge were calculated. BLEU score for SMT system with a spatial knowledge was slightly higher (28.0 vs 28.3) but the difference is not significant. This can be explained so that general purpose development and evaluation corpora which were used in evaluation do not contain many ambiguous geographical names. Evaluation using task specific development and evaluation corpora needs to be performed later in a course of the project.

A ranking of translated sentences relative to each other was used for manual evaluation of systems. This was the official determinant of translation quality used in the 2009 Workshop on Statistical Machine Translation shared tasks [16].

A web based evaluation environment where we can upload sources sentences and outputs of two MT systems as simple txt files was developed. Once evaluation of two systems is set up we can send a link of evaluation survey to evaluators. Evaluators are

³ The DGT Multilingual Translation Memory of the Acquis Communautaire (<http://langtech.jrc.it/DGT-TM.html>)

⁴ The monolingual training data from the Fourth Workshop on Statistical Machine Translation (<http://www.statmt.org/wmt09/translation-task.html>)

evaluating systems sentence by sentence. Evaluators see source sentence and output of two MT systems. The order of MT system outputs in evaluation differs; sometimes evaluator gets the output of the first system in a first position, sometimes he gets the output of the second system in a first position. Evaluators are encouraged to evaluate at least 25 sentences, we allow evaluator to perform evaluation in small portions. Evaluator can open the evaluation survey and evaluate few sentences and go away and come back later to continue. Each evaluator never gets the same sentence to evaluate. We are calculating how often users prefer each system based on all answers and based on comparison of sentences.

When we calculate evaluation results based on all answers we just count how many times users chose one system to be better than other. In a result we get percentage showing in how many percents of answers users preferred one system over the other. To be sure about the statistical relevance of results we also calculate confidence interval of the results. If we have A users preferring the first system and B users preferring the second system, then we calculate percentage using Eq. (7) and confidence interval using Eq. (8).

$$p = \frac{A}{A+B} 100\% \quad (7)$$

$$ci = z \sqrt{\frac{p(1-p)}{A+B}} 100\% \quad (8)$$

where z for a 95% confidence interval is 1.96.

When we have calculated p and ci , then we can say that users prefer the first system over the second in $p \pm ci$ percents of individual evaluations. We say that evaluation results are **weakly sufficient** to say that with a 95% confidence the first system is better than the second if Eq. (9) is true.

$$p - ci > 50\% \quad (9)$$

Such evaluation results are weakly sufficient because they are based on all evaluations but they do not represent system output variation from sentence to sentence. We can perform system evaluation using just one test sentence and get such weakly sufficient evaluation results. It is obvious that such evaluation is not reliable. To get more reliable results we have to base evaluation on sentences instead of all answers. We can calculate how evaluators have evaluated systems on a sentence level; if we have A evaluators preferring the particular sentence from the first system and B evaluators preferring sentence from the second system, then we can calculate percentage using Eq. (7) and confidence interval using Eq. (8). We say that particular sentence is translated better by the first system than by other system if Eq. (9) is true. To get more reliable evaluation results we are not asking evaluators to evaluate sentences which have sufficient confidence that they are translated better by one system than by other. When we have A sentences evaluated to be better translated by the first system and B sentences evaluated to be better translated by the second system or systems are in tie, then we can calculate evaluation results on sentence level using Eqs. (7) and (8) again. And we can say that evaluation results are **strongly sufficient**

to say that the first system is better than the second in the sentence level if Eq. (9) is true. We can say that evaluation is just **sufficient** if we ignore ties.

Table 6. Manual evaluation results.

System1	System2	Language pair	p	ci
SMT Spatial	SMT baseline	English-Latvian	58.67 %	±4.98 %

The baseline SMT system was compared to the SMT system with the spatial knowledge using system for manual comparison of two systems described above. Results of manual evaluation are given in Table 6. Manual comparison of English-Latvian factored and baseline SMT systems shows that evaluation results are sufficient to say that spatial SMT system is better than baseline system, because in 58.67% ($\pm 4.98\%$) of cases users judged its output to be better than the output of baseline system.

4. Current Results and Future Work

This paper presents ongoing research. SOLIM spatial ontology language used in this project is specified and implemented. But it is in evaluation stage now; it might be altered depending on evaluation results. Use of SOLIM language in machine translation system is one of its' evaluation scenarios. Spatial ontology containing information about all continents, countries, USA states, world's major cities and all populated places in Latvia and Lithuania has been implemented and tested. There is a reasoner created for SOLIM ontology. SMT training and translation processes have been adapted so that SMT system can use reasoner to annotate SMT training data and to preprocess input sentences. Test SMT system has been trained, and it is ready for initial evaluation and elaboration.

Current implementation of ontology integration uses all spatial relations which we can get from the reasoner, but it is obvious that different relations have different impact on translation quality. Impact of each spatial relation needs to be evaluated and optimal combination needs to be found.

Spatial information is just one type of semantic knowledge which can be added to SMT system. Enriching SMT with other types of semantic knowledge coming from other types of ontologies is also a perspective research direction.

Acknowledgments

The research within the project SOLIM leading to these results has received funding from the European Union EUREKA's Eurostars Programme, grant agreement no E! 4365 and this research was partially supported by the European Social Fund (ESF) activity No. 1.1.2.1.2 "Support to doctor's studies", project No. 2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004. I would like to thank Karlis Goba, Tatiana Gornostay, Thomas Dohmen and Mark Vreijling for contributing to parts of this paper.

References

- [1] Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R.: The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311 (1993).
- [2] Koehn, P., Och, F. J., Marcu, D.: Statistical phrase based translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL)* (2003)
- [3] Koehn, P. and Hoang, H.: Factored Translation Models. In *Proceedings of EMNLP'07* (2007).
- [4] Chiang, D.: Hierarchical Phrase-Based Translation. In *Computational Linguistics* 33(2), pp. 201-228 (2007)
- [5] Marcu, D., Wang, W., Echiabi, A., and Knight, K.: SPMT: statistical machine translation with syntactified target language phrases. In *Proc. of the 2006 Conference on Empirical Methods in Natural Language Processing* (Sydney, Australia, July 22 - 23, 2006). *ACL Workshops. Association for Computational Linguistics*, Morristown, NJ, pp. 44-52. (2006)
- [6] Li, Z., Callison-Burch, C., Dyer, C., Ganitkevitch, J., Khudanpur, S., Schwartz, L., Thornton, W., Weese, J., Zaidan, O.: Joshua: An Open Source Toolkit for Parsing-based Machine Translation. In *Proceedings of the Workshop on Statistical Machine Translation (WMT09)*, (2009)
- [7] Quirk, C., Menezes, A., Cherry, C.: Dependency Treelet Translation: Syntactically Informed Phrasal SMT, In *Proc. of ACL 2005* (2005)
- [8] Dohmen T. et al.: D2.1 Language Design, SOLIM project deliverable (2009)
- [9] Randell, D. A., Cui, Z. and Cohn, A. G.: A spatial logic based on regions and connection, *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*, pp. 165–176, Morgan Kaufmann, San Mateo (1992)
- [10] Egenhofer, M. J. and Franzosa, R.: Point-Set Topological Spatial Relations. *International Journal of Geographical Information Systems*, Vol. 5(2), pp. 161–174 (1991)
- [11] Katz, Y. and Cuenca Grau, B.: Representing Qualitative Spatial Information in OWL-DL. In *Proc. of 1st OWL: Experiences and Directions Workshop (OWLED2005)*, Galway, Ireland (2005)
- [12] Koehn, P., Federico, M., Cowan, B., Zens, R., Duer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pp. 177-180, Prague (2007)
- [13] J. Tiedemann, News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces, in N. Nicolov, K. Bontcheva, G. Angelova, R. Mitkov (eds.) *Recent Advances in Natural Language Processing* (vol V), 237-248, John Benjamins, Amsterdam/Philadelphia (2009)
- [14] Papineni, K., Roukos, S., Ward, T., Zhu, W.: BLEU: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL)* (2002)
- [15] Snover, M., Madnani, N., Dorr, B.J., Schwartz, R.: Fluency, adequacy, or HTER?: exploring different human judgments with a tunable MT metric, *Proceedings of the Fourth Workshop on Statistical Machine Translation*, Athens, Greece (2009)
- [16] C. Callison-Burch, P. Koehn, C. Monz, J. Schroeder, Findings of the 2009 Workshop on Statistical Machine Translation, in *Proceedings of the Fourth Workshop on Statistical Machine Translation*, 1-28, Athens, Greece (2009)

Domain Specific Languages and Tools

This page intentionally left blank

Practitioners View on Domain Specific Business Process Modeling

Janis BICEVSKIS, Jana CERINA-BERZINA, Girts KARNITIS, Lelde LACE,
Inga MEDVEDIS and Sergejs NESTEROVS
University of Latvia

Abstract. Practitioners view on modeling with domain specific languages is presented in this paper. It is shown, that unlike general-purpose modeling languages, domain specific languages provide means for concise representation of semantics of the particular business domain, enabling development of consistent and expressive business process models. Resulting models can be used not only as specifications for information systems, but also for generation of implementation artifacts, creating documentation, testing and for other purposes. Thus one of the most principal goals of model driven architecture – the development of model-based information system – is achieved.

Keywords: Business Process Modeling, BPM, Domain Specific Languages, DSL, Model Driven Architecture, MDA.

Introduction

A perfect information system, which is consistent with all customer requirements, reliable and flexible, still remains only a dream for IT developers. Main obstacle in the way to this dream is inability of customers, who do not know information technologies in details, to define their requirements clearly and to communicate them to the developers. Traditionally, information systems have been developed in compliance with some standardized documentation, for example, software requirements specifications, but, in practice, requirements, if they are formulated in natural language, tend to be inaccurate and ambiguous. The situation is worsened by often changing customer requirements. All this places software developers in unenviable situation – they must develop software according to inaccurate and changing specifications.

One possible solution is to use a formal language to define requirements and to make a model for the system (UML [1], BPMN [2]). This can eliminate ambiguity of requirements, and can enable direct translation of specification into application. Thus, the problem of changeability of requirements becomes resolvable – changes can be introduced into model, and then application can be automatically generated.

However, despite decades-long efforts, these problems still are not completely solved. Traditional CASE technologies have given only partial results (see, for example, Oracle Designer [3]). Fierce competition in IT market demands information systems of exceptional quality, and support from traditional CASE technologies is not sufficient to provide adequate user interface, high usability, maintainability, performance... Flexibility against changing requirements is still limited. For example, if requirements are changed, and these changes cannot be represented in the formal

specification (as specification language is not sufficient), they must be incorporated directly into the source-code of the system, and, in case of automatically generated source-code serious problems can arise. CASE tools are also quite conservative and slow to catch up the fast development of the programming technologies. As a result, only a fraction of applications can be generated from specifications.

IT experts are still looking for new ideas. One of the recent developments is Model Driven Architecture (MDA [4, 5]). In order to make application development more flexible, this approach splits the process into two steps. First, the platform-independent model (PIM) is made in some general-purpose or domain-specific modeling language (DSL), for example, UML. Second, PIM is translated to a platform-specific model PSM, thus obtaining an executable application. This separation allows for more flexible generation of applications and development of user-friendly information systems. MDA approach evolves very fast, but some challenges are also at a glance.

General-purpose modeling languages, including UML, often used to make PIMs, are difficult to grasp for non-IT professionals, the future users of information systems. Even if they read and accept the models, their understanding is not deep enough, and they undervalue consequences of decisions behind these models. Code, generated from the PSMs, still does not produce usable and reliable software. Information systems are not flexible, and it is hard to achieve compliance with the models. If changes are made directly in the generated code, consequent re-generation can void them.

Trying to follow the path of MDA often ends with UML specifications, that is just one of the sources of information for developers of the software. Only in some projects UML models are directly translated into software ([6]).

Facing these problems in the practice again and again, and having advanced tool-building platform ([7]) at hand, we tried to solve them building domain-specific business process modeling tools, according to the following principles:

1. Tools must be comprehensible to non-IT specialists, because modeling will be done mostly by domain professionals.
2. Specific requirements of business domain and of information system development must be taken into account.
3. Modeling of real-life situations in the business domain must be possible, as well as ensuring and showing to users that information systems treat these situations correctly.

We have made a graphical domain-specific language, which, we hope, can easily be understood by non-IT professionals. This language is domain-specific: first, it can be used only for modeling of business processes, and, second, it can be used only in specific organizations. The language, called ProMod, is extended subset of BPMN (Section 1). Language deals mostly with behavioral aspects and do not try to cover entire enterprise architecture as for example ArchiMate ([8]) do. Key feature of ProMod is that semantics of graphical primitives is deeply specific for organizations where it is intended to be used. Business process model is a set of diagrams, interconnected with tree-like structures of enterprise data. Unlike many general-purpose modeling tools ensuring only syntactical correctness of models, ProMod provides tools for checking semantic consistency and completeness with domain-specific rules – this kind of validation is impossible in a general-purpose language.

Narrow usage domain of ProMod raises a question of amortization of efforts, i.e. whether the benefits gained are worth the time and money spent developing the language. Using transformation driven architecture to build DSL tools can solve this problem. We used metamodel-based graphical tool-building platform GrTP and it

enabled us to create both graphical editor and consistency checker in a reasonably short time. This paper deals mostly with ProMod DSL – for detailed discussion of tool-building platform GrTP see [7], [9] and [10].

The paper is organized as follows. Section 1 contains description of our business domain and main features of ProMod DSL. Section 2 is a brief introduction to tool-building platform GrTP. Section 3 discusses current usage and ideas about future development of ProMod DSL.

1. Features of Domain-specific Modeling Language

General-purpose modeling languages offer a fixed set of primitives: objects, attributes, connectors etc., with predefined semantics, that cannot be extended, or can be extended in some limited predefined manner (as, for example, stereotypes and profiles in UML [11], [12] or styling of graphics and option to adding attributes in BPMN [2], [13]). Such notation is suitable for modeling in general, still much of the domain-specific information remains outside the models. In the domain-specific languages we are looking for ways to extend the set of modeling primitives with ones, specific to the particular business domain ([14]).

1.1. Modeling Domain: State Social Insurance Agency

The domain, where we are looking for specific concepts, repetitive patterns and clichés of business organization to enrich the modeling language, is Latvian State Social Insurance Agency (SSIA) – a government institution, providing pensions, benefits, allowances etc. Like in many government institutions, all activities in SSIA are strictly prescribed by legislation and local instructions, but, unlike most government institutions, SSIA is a client-oriented enterprise – its main function is to service clients. Servicing clients in a vast majority of cases means processing documents: a client claims for some social service and provides appropriate documents, these documents go thru a workflow, and in the end approval or rejection letter is sent to the client.

The domain of social security is sensitive sphere in Latvia and it is a target of frequent political decisions and new regulations resulting in frequent changes of information systems.

SSIA has recognized need for the management of business processes. Many of the business processes have already been defined in richly annotated IDEF0 ([15]) diagrams. Currently SSIA is planning to include business process models into the instructions for the stuff and to use them as essential part of the requirement specifications for the information systems. In the same time, as the numbers of diagrams is constantly growing, and the diagrams are independent VISIO files, it becomes harder and harder to keep them consistent.

1.2. Overview of Language ProMod

ProMod is based on a subset of BPMN and keeps its more frequently used graphical symbols: activities, events, sequence and message flows, data objects etc. These symbols sometimes have specific semantics for SSIA. For example, occurrence of an event means, that a person or an organization has brought a package of documents – events are also color-coded to show whether they originate within SSIA or come from

another institution. Message and sequence flows always carry documents and are marked as significant (bold arrows) or insignificant, and so on.

Graphical symbols have rich set of attributes. Activity, for example, in addition to traditional attributes (name, textual description, performer...) has also domain-specific attributes (regulations defining it, document templates involved, customer services fulfilled...) and modeling process attributes (acceptance status, error flags, version...).

A model in ProMod is a set of diagrams representing business processes (Figure 1). There are three types of diagrams (all behavioral according to UML taxonomy):

1. Business process diagrams are used to describe business processes for employees of the organization. They are simple and easy to read.
2. Information system diagrams are also used for process modeling, but are intended to describe the process in a way, suitable for development of information systems. For further differentiation between these two types of diagrams see Section 1.6.
3. Customer service diagrams provide description of the business processes from the viewpoint of services provided to the customers – see Section 1.5.

Besides these diagrams, structured lists are also part of the model, representing:

1. Structure of organization,
2. Regulations and local instructions defining the business processes,
3. Information artifacts – documents and pieces of information,
4. Services provided to customers.

These lists are not only convenient way to enter values of attributes – they by themselves carry essential data, establish connection between various fragments of model and are used for consistency checking and reporting.

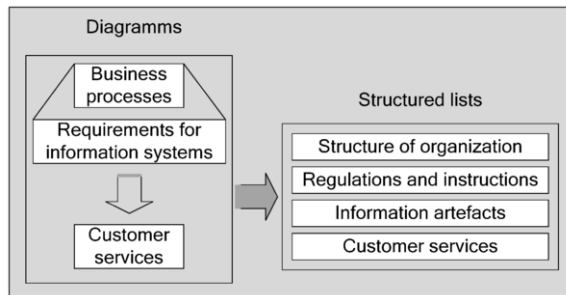


Figure 1. Diagrams and structured lists

1.3. Refinement of Business Process Models with Enterprise Information

Information stored in the structured lists is needed not only for modeling of business processes – in fact it is the basic enterprise information. In most organizations this information can be found in enterprise information systems (EIS), but unfortunately these systems have not been built with business process modeling in mind. Traditional modeling tools often are incapable to connect to these data bases and obtain the data. Models, therefore, remain isolated from the real world, and if, for example, enterprise information is changed, it is easy to forget to change the models accordingly.

In domain-specific modeling tools it is natural to provide means for data exchange with EISs. ProMod provides these means, giving so the following advantages:

- Enterprise information can easier be maintained in EIS – there the information is connected to another data, quality can be checked and responsibility for maintenance can be assigned.
- Information is not duplicated, if modeling tools take it from EIS.
- Business models are closely connected to real life of the organization, and the risk for them to become outdated and inadequate is smaller.
- If needed, it is possible to integrate business process models into information systems, to show step-by-step progress of the instance of business process.
- It is possible to analyze business processes in context of enterprise data, showing, for example, which other regulations and which steps of business processes will be affected, if some part of regulation is changed (that is important in SSIA, because of frequent and voluntary changes in regulations).

In addition it is possible to interconnect different diagrams and graphical symbols and to make new types of diagrams according the domain-specific logic. Customer service diagrams, for example, consists of action symbols, defined in other diagrams, and joined together, because they are needed for particular customer service (as mentioned in Section 1.2, customer services are part of enterprise data).

1.4. Domain-Specific Consistency Rules for Business Processes

Important aspect of modeling is consistency of created models. According to the scope consistency can be:

1. In the level of element, for example, whether all mandatory attributes have been entered.
2. In the level of diagram, for example, whether diagram begins with an event.
3. In the level of model, for example, whether all referenced sub-processes are defined somewhere.

Above mentioned are universal consistency rules, but in ProMod, rules, reflecting specificity of SSIA are more essential. As the main goal of business processes is to process customer documents, it must be checked, whether all documents, provided by customer, are used in some step. It must be checked, whether set of documents from event starting the business process match to set of documents used in decomposition of its first step. It must be checked, whether all steps in all business processes are needed for some customer services, and so on.

As enterprise data is linked to the model, it is possible to check, whether performer of the step still exists in SSIA, whether organizational structure of SSIA have not been changed, whether regulations, defining business process, have not been expired... As the possibilities of domain-specific consistency checking seem to be unlimited, ProMod provides means for easily adding new rules.

In ProMod consistency checking is not performed during creation or modification of the diagrams – consistency check is a separate action, and inconsistent models can be stored in the system and kept for a while. This approach gives some benefits:

1. It is possible to start from rough sketches made by insurance professionals of SSIA, work with them and in the end turn them into consistent models.
2. This approach is more suitable for non-IT professionals, because they tend to concentrate on the main ideas and think, that consistency details are boring.

1.5. Business Processes and Customer Services: Two Views on the Same Model

At the very first steps of the business process modeling, when trying to identify and name business processes, there are two essentially different options. First, we can look at the organization from management's perspective and classify processes according to the way they are performed. Second, we can take perspective of customers and classify processes according to services or products they are producing.

Management's perspective seems more natural, and has been chosen in SSIA. We believe that it will be the case in most government institutions. If one reads statutes of SSIA, the first higher level business processes are obviously the functions mentioned there: grant social insurance services, provide consultations, register socially insured persons, etc. (see ProMod business process diagram in Figure 2.a). Customer's perspective would lead to business processes related to the customer services: grant retirement pension (including consultations, registration and everything else), grant disability pension or grant childbirth benefit.

If, following the management's perspective, we perform top-down decomposition of high-level abstract business processes; we see that many steps are independent of the customer services they provide. For example, steps "Receive documents", "Register documents" and "Send resolution" (Figure 2.b) are almost the same whether request for retirement pension or disability pension is being processed. Differences in processing various customer services show up only in deeper level of decomposition (Figure 2.c).

Essential drawback in taking management's perspective is that in the resulting models customer services are not clearly represented – they are "dissolved" in detailed lower-level diagrams. Customer, for example, is always interested in one service at a time and business process diagrams are not helpful to him.

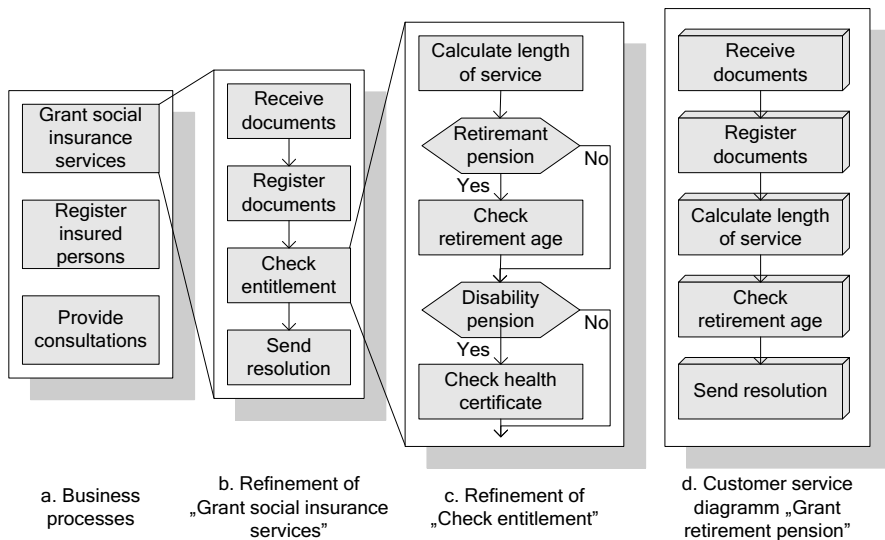


Figure 2. Business processes and customer services

To resolve this contradiction between management's and customer's perspectives, in ProMod we have introduced special type of diagrams, called Customer Service Diagrams. These diagrams are made for every service provided, and contain all steps

from various business process diagrams, needed to provide that service (Figure 2.d). This is a distilled value chain for one particular customer service. These diagrams do not contain any new information they are just a different views to business process diagrams, and in our editor they are made semi-automatically. Customer Service Diagrams bear some resemblance to use-cases and communication diagrams in UML.

1.6. Modeling for Humans and Modeling for Information Systems

The MDA approach encourages automatic translation of business process models into implementation artifacts (database objects or executable code). In the situation when both restructuring of the business operations and development of an information system are the goals of the business process modeling, it is tempting to assume, that the same set models can be used for both purposes. This point seems even stronger, because the same modeling language can be used to pursue both goals. However models, that can be easily read by employees, lack accuracy and detail needed for development of information systems, but models, suitable for development of information systems, are much too detailed and boring, to be read by employees. The difference is not only in the degree of elaboration: style, graphic representation, cultural biases and even human ambitions must be taken into account.

We faced all these challenges in SSIA, where business divisions were responsible for business processes, but development of information systems were split into separate department and partially outsourced. For this reason there are two visually different diagram types in ProMod: one intended for employees, and other – for development of information systems (Figure 3).

Diagrams for employees have limited set of graphical symbols; they are intended to specify sequence of steps, rather than detailed logic; and, in order not to disturb employees of business divisions, they look much like previously used IDEF0 specifications. Level of detail is acceptable, if knowledgeable insurance professionals have no difficulties to follow the business process, using them.

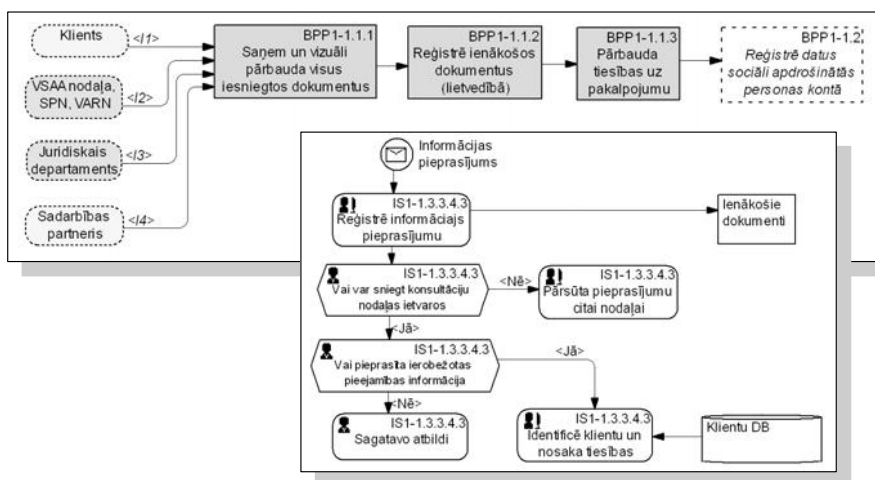


Figure 3. Diagrams for employees and for development of information system (real-life example)

Diagrams for development of information systems have richer set of graphical symbols. Level of detail is higher – professional information system designers must have no difficulties to design information system using these diagrams.

Both types of diagrams are elaborated by top-down decomposition, and higher level information system diagrams in most cases are subordinated to the lower-level business diagrams.

1.7. Using Models to Create Documentation

Essential part of the software to be created is documentation. In most cases it is a set of textual documents or help files. If some model of the system operation is described, then the model itself is shown as a picture inside the document – just a visual extra. If afterwards model of operation is changed, then the textual description is modified and the picture is replaced. Usually models are referred to in many documents: requirements specifications, design description, user guides etc. In practice it often happens, that some changes are reflected in one document, but forgotten in other, yielding to discrepancies between various documents and the system itself.

We propose that models must be used as the primary place to store information about the system, and that the documentation, at least partially, must be generated from the models. This ensures conformance of system, its model and its documentation.

We propose the following scenario for development of the system:

- Client defines his requirements in a form, convenient to him: as textual documents, sketches of models, formats of required reports and so on.
- System analysts create a model of the system containing both formal description (diagrams in the domain-specific language and formalized attributes) and informal description (including requirements given by client: documents, sketches, report formats...). The presence of informal descriptions enables automatic generation of the requirements specification document, containing both formal models and requirements of the client. Such a document is comprehensible to the client, even if pure models are not.
- Due to the rich informal part of requirement specifications, client quickly grasps the essence of the formal models and learns to read them. When client has gained some knowledge about the models, he can even begin to give his requirements in form of models (and in our practice this is often the case). The proportion of formal models grows, but informal descriptions are kept for better understanding.
- In software design phase models are further elaborated and design-specific information is added. The result is complete design model, from which design documentation can be generated (or at least most of it).
- Models of the design phase are used by software developers, ensuring that the same information is used by programmers, designers, system analysts and the client. Models can be used to generate applications, and can be interpreted by application during execution.
- Models contain valuable information for user guides too. If properly extended, they can be used to partially generate user guides, ensuring that user documentation will not be outdated by several software versions.
- If errors in system must be corrected, models are changed accordingly, and documentation is regenerated.

- Change request are supplied by a client as formal models or as informal descriptions and goes thru the above described workflow, and generation of consistent documentation is ensured.

According to this methodology the same set of models is used in all stages of software lifecycle. Of course these are not exactly the same models: initially they contain only information provided by customer, and then they are enriched and elaborated and transformed. The consequent usage of models thru the lifecycle dramatically reduce error rate when compared to use of vaguely connected set of documents for requirement specification, design and user guides. The situation is further improved, if the models are used during execution of applications.

Our approach is model-centric – the model is the central artifact used in all stages of software lifecycle, and in every stage the model is viewed from slightly different viewpoint and enriched with specific information.

2. Transformation-Driven Architecture and Tool Building Platform GrTP

Nowadays, when appropriate tool-building platforms are available, it would be unusual to make domain-specific tools from scratch. Therefore, we are using a tool-building platform which is developed in Institute of Mathematics and Computer Science of University of Latvia, and is called GrTP [7, 9, 10]. The universal functionality, which is common in many domain-specific tools, is implemented as part of the platform. Convenient interfaces are provided for adding features needed for specific tools. Using the tool-building platform, the first version of the domain-specific tool can be created in a short time and with reasonable effort. It can then be flexibly adapted to the needs of customers.

The recent version of GrTP is based on principles of the Transformation-Driven Architecture (TDA [7]). In this section, the key principles of the TDA and GrTP as well as their applications in implementation of DSLs are discussed.

2.1. Transformation-Driven Architecture

The Transformation-Driven Architecture is a metamodel-based approach for system (in particular, tool) building, where the system metamodel consists of one or more interface metamodels served by the corresponding engines (called, the interface engines). There is also the Core Metamodel (fixed) with the corresponding Head Engine.

Every engine is responsible for providing some specific functionality to the end-user (for example, a possibility to work with graphical diagrams). Every engine is working only within the boundaries of its corresponding metamodel. That kind of independence allows one to develop and test engines separately thus improving the quality of software. Engines can work together because of their need to support so called “Events – Command” interconnection mechanism, defined by the Head engine. Events provide information about actions of the user, which are stored as instances of the metamodel of the particular engine. Commands provide information about changes in metamodel and requested actions.

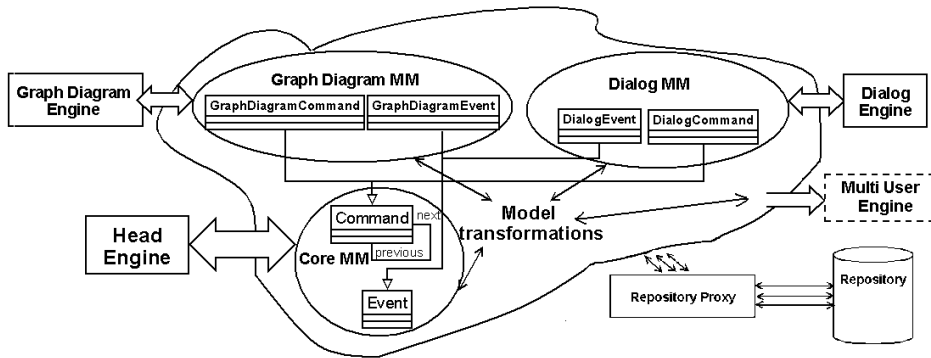


Figure 4. Transformation-driven architecture framework filled with some interfaces

Since every engine works only within the boundaries of its metamodel, some tools for connecting these metamodels are needed. Model transformations are used for this purpose. Moreover, using of metamodel transformations is the second basic principle of TDA. Transformations are processing events created by engines. They accomplish changes in metamodel and data and create commands that will subsequently be executed by engines. For the execution of transformations to be successful, it is often needed to add new classes and associations to the metamodel.

It is also possible to connect external engines, which either do not have interface metamodel, or stores only partial data in their metamodel. These engines are called by model transformations using one-way calls.

2.2. TDA-Based Tool Building Platform GrTP

Using the TDA approach, we have developed a concrete tool building platform called the GrTP by taking the TDA framework and filling it with several interfaces. Besides the core interface, two basic interfaces have been developed and plugged into the platform in the case of GrTP:

- The graph diagram interface is perhaps the main interface from the end user's point of view. It allows user to view models visually in a form of graph diagrams. The graph diagram engine [7] embodies advanced graph drawing and layouting algorithms [16] as well as effective internal diagram representation structures allowing one to handle the visualization tasks efficiently even for large diagrams.
- The property dialog interface allows user to communicate with the repository using visual dialog windows.
- When building domain-specific modeling language for SSIA, it was necessary to create two external engines, which do not use events and commands, but are called by means of model transformations:
- Multi-user engine is based on information of project and graph diagrams and ensures that many users can work together with the same model (one common model on the server and separate local models for different users). Only one user is allowed to edit a diagram at any given moment. Multi-user engine uses its metamodel to save information about correspondence of server and local models.

- Microsoft Word engine is completely external module. It generates Word documents according to data of the metamodel by calling model transformations.

The final step is to develop a specific tool within the GrTP. This is being done by providing model transformations responding to user-created events. In order to reduce the work of writing transformations needed for some concrete tool, we introduce a tool definition metamodel (TDMM) with a corresponding extension mechanism. We use a universal transformation to interpret the TDMM and its extension thus obtaining concrete tools working in such an interpreting mode.

2.3. Tool Definition Metamodel

First of all, let us explain the way of coding models in domain specific languages. The main idea is depicted in Figure 5. The containment hierarchy *Tool* → *GraphDiagramType* → *ElementType* → *CompartmentType* (via base link) forms the backbone of TDMM. Every tool can serve several graph diagram types. Every graph diagram type contains several element types (instances of *ElementType*), each of them being either a box type (e.g., an *Action* in the activity diagram), or line type (e.g., a *Flow*). Every element type has an ordered collection of *CompartmentType* instances attached via its base link. These instances form the list of types of compartments of the diagram elements of this type. At runtime, each visual element (diagrams, nodes, edges, compartments) is attached to exactly one type instance.

The extension mechanism is a set of precisely defined extension points through which one can specify transformations to be called in various cases. One example of a possible extension could be an “elementCreated” extension providing the transformation to be called when some new element has been created in a graph diagram. Tools are being represented by instances of the TDMM by interpreting them at runtime.

Therefore, to build a concrete tool actually means to generate the appropriate instance of the TDMM and to write model transformations for extension points. In such a way, the standard part of any tool is included in the tool definition metamodel meaning that no transformation needs to be written for that part.

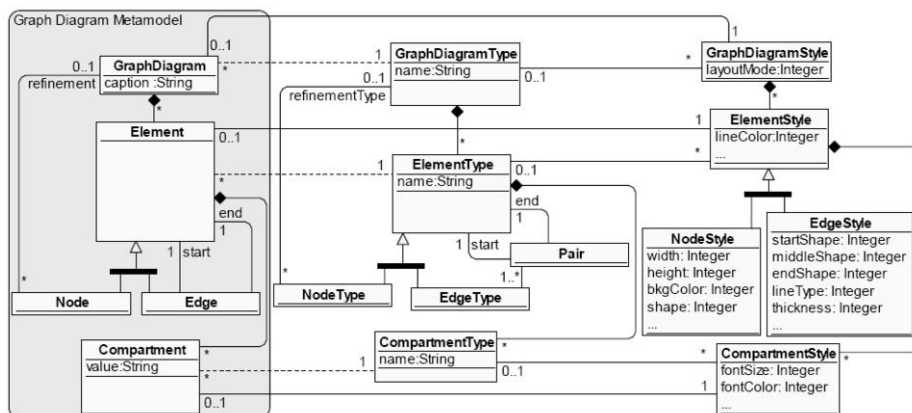


Figure 5. The way of coding models

3. Some Applications of Domain-Specific Models

Business process modeling is not an end in itself – models are built to make high-quality and convenient information systems. Sensitivity of social security and frequent changes in regulations (see Section 1.1) require high reliability, flexibility and maintainability of software. Traditional method of software developments have been used for years and have not yielded desired results. Using specifications in natural language, it was impossible to achieve needed accuracy and unambiguity. We have proposed modeling with domain-specific graphical language ProMod as a solution. Applications of the modeling that are the most urgent in SSIA are described below.

Availability of models to the wide spectrum of users. Concise description of business processes in graphical diagrams can be used as instructions for employees providing customer services and as information for clients, showing what will be done in SSIA, in order to serve their requests. The best way to spread the models is to make them available on the internet.

ProMod can export diagrams, corresponding information from structured lists and descriptive documents to Web pages. Though not every diagram is suitable for every reader, and models with varying level of details and models from different perspectives must be built – for example in-depth description for employees of SSIA and simplified version for customers.

Job descriptions for SSIA employees. Job responsibilities for many SSIA employees in fact are defined by activities in the business process models – in ProMod Customer Service Diagrams are especially designed to show this. Business process diagrams must be used in job descriptions to make them more concise and easy to read compared to textual instructions. We have conducted survey, which shows ([6]), that 90% of employees in government institutions prefer graphical descriptions to textual. We believe that job descriptions for most of the SSIA employees will be covered by Customer Service Diagrams.

Software requirement specifications. Contradiction between inaccurate and changeable requirement specifications, defined in natural language, and need for high-quality information systems is well-known and has already been discussed in this paper. We believe that domain specific business modeling (especially, using ProMod Information System Diagrams) will largely improve situation in SSIA.

Conversion from models to applications. We believe that approach: “Less technical programming, more concise specifications”, and development of information systems without technical programming can become possible in the nearest future. Modeling is the first and mandatory stage in this process. In order to use information from business models in applications (no matter, whether they are generated from models or coded manually), it is necessary to transfer this information automatically or manually from repository of the modeling tool into application database. Manual transferring involves re-entering information about objects and their connections, and linking this information to the application data. This is a monotone and quite error-prone job. Transferring information with automated tools would be more efficient. This kind of transformation can be implemented with minor resources, if the modeling tool provides application programming interface to access its repository. So application can work according to models created in graphical language, but its quality (usability, reliability, security, performance etc.) remains independent of capacity of some hypothetical generator to generate a high-quality applications.

This approach has been tested in a number of medium-size projects [6], where information systems are less complex than in SSIA. In SSIA this is a next step to be taken. This approach has proved noticeable viability, and attitude of users towards the graphical models as requirement specifications and as core of user guides was surprisingly positive. Users considered graphical diagrams as highly comprehensible and soon gave up reading thick and boring manuals. Admittedly this approach asks for further development, but we see this as a realistic way to develop user-friendly, flexible and reliable information systems.

Formal model as testing model. In software testing the model is often emphasized, according to which testing of the system begins already in requirements specification phase by accumulating test cases for proving compliance of software and specifications. If the formal MDA model can be built by system analysts from use cases, the use cases must contain information needed to test, whether the resulting system operates correctly. Every use case describes sequence of action during some operation, and naturally these actions are used for testing of software, developed from these models. This approach is popular in practice.

Though admittedly the testing based on single use cases can be insufficient for high software quality. Essentially higher quality of testing can be achieved using model-driven testing approach. In theory of testing the control flow graph is well-known: actions are vertices, but transitions are edges. One usage of the system is one path thru the graph. Testing of the program is said to be complete, if all transitions (edges) are traversed during some test case (criterion C1).

This approach is suitable also for business processes and other behavioral models. The system can be considered as sufficiently tested, if all possible sequences of actions (according to the model of the system) are tested. As testing according to C1 can be too laborious, testing sometimes is restricted to use cases accumulated during requirement specification phase, but this is clearly much weaker approach.

4. Conclusions

The following conclusions can be made from our experience with creating domain specific language ProMod and with business process modeling in SSIA:

- Business process modeling with domain-specific language is preferable, compared to modeling with general-purpose language.
- Domain specific models have wide application: they can be used as core of job descriptions and requirement specification, as source of information for automatic generation of applications, as testing model for model driven testing.
- With tool building platform GrTP domain-specific languages and supporting tools: graphical editor, consistency checker and model-to-application information transfer utility, can be created in short time and with modest resources.
- Business process modeling with consistency checks in a very short timeframe allows to identify contradictions and bottlenecks of processes descriptions
- Move to model-driven architecture profoundly changes information system development technology. If an information system has been developed with traditional methods, serious modifications and enormous resources can be needed.

Practical experience in business processes modeling in large and complex government institution SSIA, confirms feasibility and advantages of model-driven development of information systems.

Acknowledgments

This research is partly supported by European Social Fund.

References

- [1] UML, <http://www.uml.org>.
- [2] BPMN, <http://www.bpmn.org>.
- [3] Oracle Designer, <http://www.oracle.com/technology/products/designer/documentation.html>
- [4] MDA Guide Version 1.0.1. OMG, <http://www.omg.org/docs/omg/03-06-01.pdf>.
- [5] Flore, F.: MDA: The Proof is in Automating Transformations between Models. OptimalJ White Paper. <http://www.dsic.upv.es/~einsfran/mda/modeltransformations.pdf>
- [6] Cerina-Berzina, J., Bicevskis, J., Karnitis, G.: Information systems development based on visual Domain Specific Language BiLingva. In: Preprint of the Proceedings of the 4th IFIP TC 2 Central and East Europe Conference on Software Engineering Techniques, CEE-SET 2009, Krakow, pp. 128-137. (2009)
- [7] Barzdins, J., Cerans, K., Kozlovics, S., Rencis, E., Zarins, A.: A Graph Diagram Engine for the Transformation-Driven Architecture. In: Proceedings of the IUT'09 Workshop on Model Driven Development of Advanced User Interfaces, pp. 29-32, Sanibel Island, USA (2009)
- [8] Lankhorst, M., et al.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer (2009)
- [9] Barzdins, J., Zarins, A., Cerans, K., Grasmanis, M., Kalnins, A., Rencis, E., Lace, L., Liepins, R., Sprogis, A., Zarins, A.: Domain Specific languages for Business Process Management: a Case Study. In: Proceedings of DSM'09 Workshop of OOPSLA 2009, Orlando, USA (2009)
- [10] Barzdins, J., Kozlovics, S., Rencis, E.: The Transformation-Driven Architecture. In: Proceedings of DSM'08 Workshop of OOPSLA 2008, pp. 60—63, Nashville, USA (2008)
- [11] Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall (2004)
- [12] Weiklens, T.: Systems Engineering with SysML. Morgan-Kaufman OMG Press (2007)
- [13] White, S.A., Miers, D., Fischer, L.: BPMN Modeling and Reference Guide. Future Strategies Inc. (2008)
- [14] Lan Cao, Balasubramaniam Ramesh, Matti Rossi: Are Domain-Specific Models Easier to Maintain Than UML Models? In: IEEE Software, vol. 26, no. 4, pp. 19--21 (2009)
- [15] ICAM Architecture Part II - Volume IV - Function Modeling Manual (IDEF0). <http://handle.dtic.mil/100.2/ADB0624>
- [16] Freivalds, K., Kikusts, P.: Optimum Layout Adjustment Supporting Ordering Constraints in Graph-Like Diagram Drawing. In: Proceedings of The Latvian Academy of Sciences, Section B, vol. 55, No. 1, pp. 43–51, Riga (2001)

The Design of Electronic Service Process Using YAWL and CPN Tools

Peteris STIPRAVIETIS^{a,1} and Maris ZIEMA^a

^a*Riga Technical University, Faculty of Computer Science and Information Technology,
Institute of Computer Control, Automation and Computer Engineering*

Abstract. The article discusses the solution of common business process design-time problems using YAWL. The approach proposed by the authors is based on the creation of business process in the YAWL environment in order to simulate the process model which could resolve some of the design-time problems as well as provide hints to correct initial process. Simulation is based on transformation to colored Petri net – the net is then simulated to identify problems such as possible infinite loops, bottlenecks, process waits and others. The article also describes technique to acquire the primitive description of process from the YAWL workflow. The primitive description is represented as oriented graph and is used to transform the YAWL workflow to another hierarchic language, in this case BPEL.

Keywords. YAWL, BPEL, simulation, transformation

Introduction

E-services are common in information society nowadays, and even though they tend to become more and more accessible and varied, the problems that occur during the design phase of the service remain the same. These problems include, for example, questions on how to facilitate the creation of business process to the user with no specific programming skills, how to define the process in a way that creates the process description abstract but accurate enough at the same time, how to check the created model – to determine the weaknesses, perform the measurements based tuning, and others. The validation of the process is even more important when the process being changed is already deployed and used in production environment – one must make sure that the changes are implemented correctly, that the new instances of the process can execute together with old instances already running. The implementation and validation of changed process also have to be simple and cost-effective enough – hence the conclusion that the solutions of these problems rely heavily on the choice of the language used to describe the process – does it provide the possibilities to validate and simulate the process.

Existing business process modeling languages can be divided in two groups. The languages of the first group are favored by the academic community, but rarely used in real-life solutions. These languages are based on Petri nets, process algebra; they have formal semantics, which allow the validation of the models described by these

¹ Riga Technical University, Faculty of Computer Science and Information Technology, Institute of Computer Control, Automation and Computer Engineering, Meza 1/3, 3rd floor, LV-1048, Riga, Latvia; E-mail: peteris.stipravietis@rtu.lv

languages. The languages of the second group are used in real-life projects much more than in academic researches. BPEL, WSFL and WSCI are among these languages. These, so called business languages, often lack proper semantics, which could lead to debate on how to interpret the business models described by these languages. The availability of different implementations of these languages from different vendors does not facilitate the situation either, yet they are used much more, compared to seldom used models described by academic languages. If a situation arises when business process model described by business language needs to be validated using Petri nets, one must either abandon the validation or transform the process model to another model, described in academic language, for example YAWL. The authors propose reverse approach – first, a process is created using academic language. The design problems of the process model can then be solved by mathematical means. Second, the verified and updated model is transformed to model described in business language. The advantages of the approach described follows:

- If a model is created using academic language, it is more readable and maintainable than the model, which is a transformation result itself. It is also easier to perform analysis of untransformed model, because the transformation could lose some design information.
- Model, transformed to business language, is already validated and ready to be executed. Of course, the model must be double-checked to make sure if it needs any corrections. The alternatives of the execution environment for the model are much more than the environments for academic languages; in addition to that, they have superior technical support.

The purpose of this article is to examine the aforementioned approach – can it be used during the design of simple e-service business project, check if it helps to resolve most common design-time problems; view the transformation from academic language YAWL to business language BPEL. The emphasis in this article is put on the simulation part of mentioned approach.

1. The Languages

YAWL stands for ‘Yet Another Workflow Language’ – the language to describe workflows. It supports non-trivial data transformations and integration of web services. YAWL is based on extended workflow nets (EWF) [1]. YAWL could be defined as Petri net, enhanced with possibility to define multiple instances, synchronizations, OR splits and joins as well as cancellation regions. The workflow definition in YAWL in hierarchically structured set of EWF nets, forming a tree. Every net consists of activities and conditions – the places and transitions of the net. Activities can be atomic or composite. In fact, every composite activity is a net itself. Every net has an input and output condition [2]. YAWL has a graphic environment in which the workflows are created and execution environment to load and run validated definitions of workflows into. YAWL-based solutions are also used in business environment, for example, YAWL4Film is a YAWL extension used in Australian Film Television and Radio School (AFTRS) [3], but extension YAWL4Health is used in Academic Medical Centre (AMC) in Netherlands [4]. YAWL is chosen as the academic language of the approach because of the following reasons:

- YAWL is based on EWF, the workflows described in YAWL can be transformed to colored Petri nets to perform simulations and formal semantic validation;
- YAWL is designed to support all the workflow patterns [5].

Another academic language is BPMN, based on process algebra, but its lack of decent simulation techniques and support of all workflow patterns made the authors choose YAWL as the academic language in proposed approach. Another academic language is BPMN, based on process algebra, but its lack of decent simulation techniques and support of all workflow patterns made the authors choose YAWL as the academic language in proposed approach.

BPEL stands for 'Business Process Execution Language'. Processes described with BPEL uses web services to exchange information with other processes or systems. BPEL is based on XML, it has not standardized graphical notation. BPEL is chosen as the business language of the approach because of the following reasons:

- BPEL is an industry standard;
- There are many popular environments available, in which to design and run BPEL processes – Microsoft BizTalk, Oracle BPEL Process Manager and IBM WebSphere among the others.

2. The Simulation

The analysis of the created business process is very important part of the design – one needs to find bottlenecks, when instance of the process or its part could use up all available resources, thus forcing other instances to wait for these resources; identify dead ends, which could lead to infinite loops and never ending process instances; find deadlocks, when process querying for the same resources effectively block each other; define fault handling and cancellation activities, which cancel all the work done by previous activities; identify reusable structures, for example, audit log activities.

Such challenges usually are overcome with the simulation. Let us inspect a certain simulation approach and see if it helps with the problems, mentioned before. The authors of the publication [6] propose simulation which uses process design data, historical data about executed process instances from audit logs and state data of the running process instances from the execution environment. Data from all three sources are combined to create simulation model – design data are used to define the structure of the simulation model, historical data define simulation parameters, state data are used to initialize the simulation model.

Altering the simulation model allows to simulate different situations, for example, to omit certain activities or divert the process flow to other execution channels. Taking into account the state data of running process instances, it is possible to render the state of the system in near future and use the information to make decisions regarding the underlying business process.

The simulation of the workflow is carried out using process data mining framework ProM [7]. To create simulation model, following steps are performed:

- Workflow design, organizational and audit log data are imported from execution environment;

- According to imported data a new YAWL workflow model is created and state data are added;
- The new model is converted to Petri net;
- Resulting Petri net is exported to simulation execution environment CPN Tools [8] as a colored Petri net.

CPN Tools environment provides the process simulation possibilities both in long and short-term, using the state data of chosen process instance. This technique differs from others (known to authors) with its degree of realism – many other methods assume that available resources are 100% dedicated to the instance being simulated and do not take into account real resource usage by other instances or processes; yet this method creates artificial delays based on historical data from audit logs and organizational model.

Returning to design-time problems, mentioned before – the authors now will evaluate the simulation approach, focusing on its capabilities to identify these problems.

A common mistake is to propose that the user of process will provide all data when prompted, for example, fill out all fields in a web form. Some fields could be left blank because user is not interested in sharing particular information (because of privacy concerns etc.) or other reasons. If a process needs such information to continue, but user does not provide it, it enters in a waiting state and theoretically may never be finished. As an example a simple YAWL workflow is provided, which expects the input of e-mail address and phone number from the user. The workflow contains AND-flow – it is finished when both branches are finished – see Figure 1.

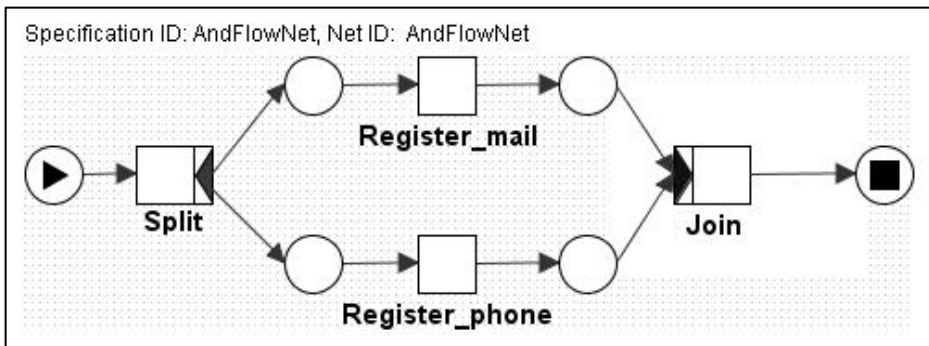


Figure 1. Simple YAWL workflow – parallel branches

The solution is to use OR-flow instead of AND, but the simulation approach discussed is not applicable – Petri nets does not support OR-flows. However, the problem can be identified by analyzing the process execution log, using, for example, ‘Basic Log Analysis’ module from ProM toolset. Figure 2 clearly shows the difference between activities executed. Naturally the question arises – why are there so many processes in waiting state and why do the users register their phone numbers far less than their e-mail addresses. As for the ‘waiting’ problem – it can be identified, but not by means of simulation.

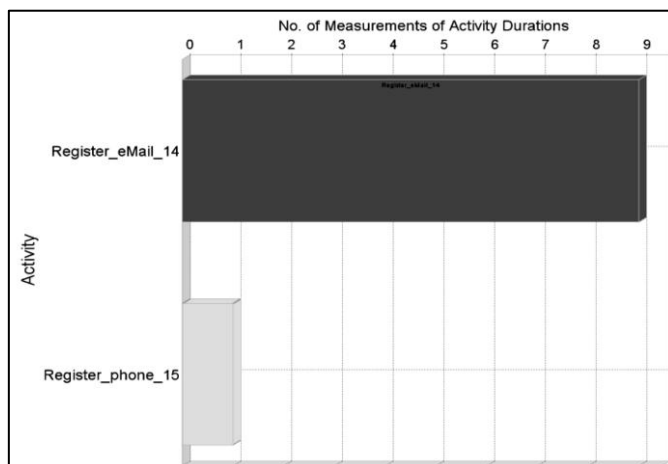


Figure 2. Difference of activities

Another task is to identify possible situations which could lead to infinite looping. The reason behind the looping mostly is incorrectly defined loop exit condition or loop variable does not have correct value assigned. This case can also be identified using ProM tools (activities within loop would be far more than others), but the simulation technique discussed can be used too.

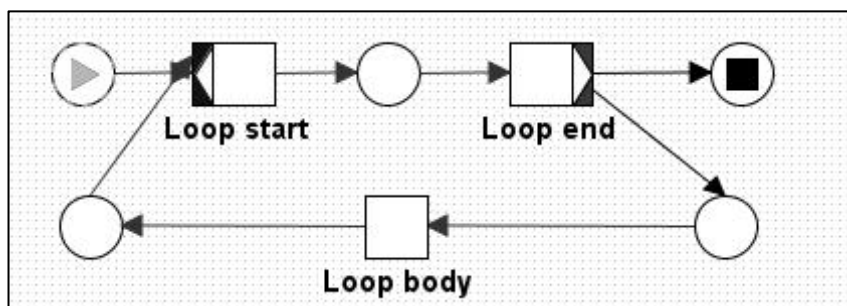


Figure 3. Simple YAWL workflow – infinite loop

Figure 3 shows the example of loop. The workflow has one local variable – ‘exitLoop’ of Boolean data type, but its value intentionally is not being changed. Figure 4 shows resulting Petri net – variables ‘loopCount’ and ‘caseId’ were added for demonstrative purposes (token colset NET_DATA is defined as product $\text{INT} * \text{BOOL} * \text{INT}$). Examining the net simulation, one observes that incoming tokens never leave loop.

The simulation to identify bottlenecks produces similar results. YAWL has mechanism to distribute activities to resources defined in its organizational model – these resources are tokens of different colorset in a colored Petri net. A transition in Petri net may fire if all the places leading to transition have tokens – if a ‘resource’ place has no tokens, transition never fires, and all ‘data’ tokens accumulate in ‘data’ place until resources are freed and transition may fire again.

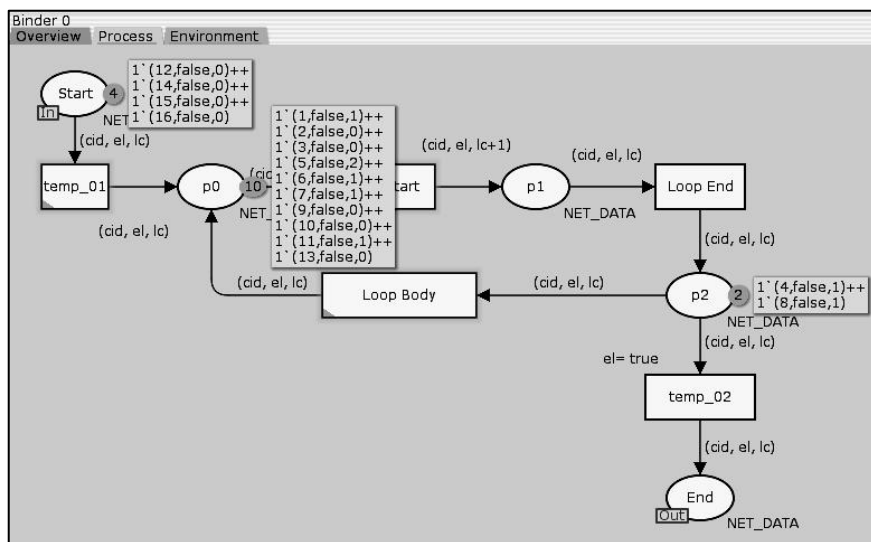


Figure 4. Colored Petri net – infinite loop

Experimenting with the proposed simulation and analyzing the results, the authors of this paper conclude that this technique could be used to solve some of the designing problems mentioned before. However, the colored Petri net generated by ProM still needs some tweaking before it can be simulated – for example, if workflow variables have not been yet initialized, ProM may produce net startup parameter file containing syntax errors. The deadlock identification is not possible, because this simulation approach does not allow multiple parallel process instances within one scope. Furthermore, the approach does not provide the cancellation simulation; hence the YAWL fault handler testing is not possible. The reason of this lack of functionality is the absence of cancellation region concept in Petri nets. Similarly, Petri nets does not support OR splits and joins (multi-choice workflow pattern, M out of N), therefore the simulation of these constructs is not possible. Such pattern is not supported by BPEL either.

3. The Transformation

When the process model has been simulated and updated accordingly, it is ready to be transformed to BPEL process. Proposed transformation at first creates primitive process structure and then creates BPEL process, using pattern recognition. Similar task - the pattern recognition and transformation to BPEL - is discussed in [9] and is based on the transformation of BPMN process to Petri net and subsequent transformation of the net to BPEL process. The approach proposed by authors of this paper uses language independent primitive structure - the notation of the process flow as a directed graph preserving the semantics of the process. The algorithms described in [9] allows to transform non-well-formed BPMN processes, using the event-action mechanism of BPEL, producing usable, albeit rather unreadable BPEL. The authors of this paper tend to transform the non-well-formed primitive structure to well-formed one (i.e. using algorithms discussed in [10]), because it would allow to transform the process to any structured language, not just BPEL.

3.1. Transition from YAWL Workflow to Primitive Structure

To facilitate the transition from YAWL workflow to BPEL process, the bipartite graph describing the workflow is simplified, resulting in primitive process structure – oriented graph with vertices of one kind. The primitive structure is created by removing the vertices standing for YAWL net places. The condition attached to the removed place is attached to new arc between the vertices of primitive structure. There is one exception, though – the end place also is included in the primitive structure.

Let us define the YAWL workflow as (P, T, W) , where:

P – finite set of places un T – finite set of transitions, $P \cap T = \emptyset$; but $W \subseteq (P \times T) \cup (T \times P)$ is set of arcs between places and transitions in such a way that no transition is connected with another transition and no place is connected with another place. $P_b \subset P$ is a subset of P and denotes end places of the workflow: $W \cap (P_b \times T) = \emptyset$. The example of simple workflow is shown in Figure 1.

Let us define primitive structure (A, W) , where:

A – finite set of activities, containing all elements from the workflow transition set T and all elements from the end places set P_b : $A = (T \cup P_b)$.

If $T_0 \xrightarrow{\text{cond}} P_n \Rightarrow T_1$ denotes a process flow from transition T_0 to transition T_1 through place P_n , when condition cond allows it, then corresponding transition from A_0 to A_1 is defined in primitive structure – $A_0 \xrightarrow{\text{cond}} A_1$.

If $T_m \xrightarrow{\text{cond}} P_n$ denotes a process flow from transition T_m to place P_n , when condition cond allows it and $P_n \in P_b$, then corresponding transition from A_m to A_n is defined in primitive structure – $A_m \xrightarrow{\text{cond}} A_n$. The example of primitive structure is shown in Figure 5.

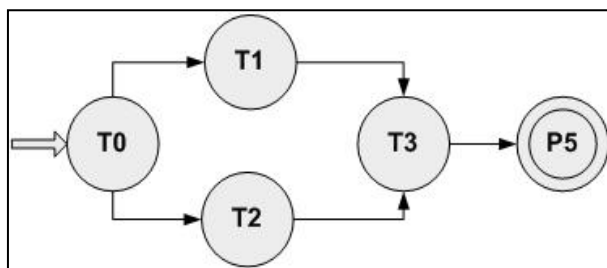


Figure 5. Primitive structure corresponding to sample YAWL workflow

The transition from YAWL workflow to primitive structure is accomplished by analyzing the YAWL workflow description in XML. Figure 6 shows the class diagram of primitive structure. Tables 1 and 2 describes the attributes and operations of the Process and Activity classes respectively.

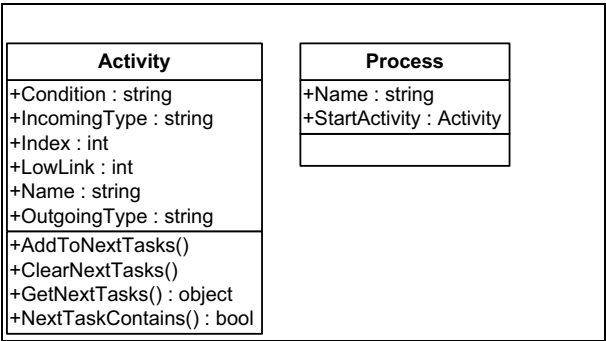


Figure 6. Class diagram of primitive structure

Table 1. Attributes and operations of Process class.

Attribute/operation	Description
Name	The name of the primitive structure
StartActivity	The starting vertex of primitive structure

Table 2. Attributes and operations of Activity class.

Attribute/operation	Description
Condition	The condition which allows the process flow through this activity
IncomingType	Type of incoming arcs: •None – no incoming arcs (for example, start activity) •Single – one incoming arc •And – many incoming arcs, synchronizing parallel flow •Xor – many incoming arcs, synchronizing exclusive flow
Index	Index – is used for cycle identification with Tarjan SCC algorithm.
LowLink	Index – is used for cycle identification with Tarjan SCC algorithm.
Name	The name of activity
OutgoingType	Type of outgoing arcs: •None – no outgoing arcs (for example, process end) •Single – one outgoing arc •And – many outgoing arcs, starting parallel flow •Xor – many outgoing arcs, starting exclusive flow
AddToNextTasks()	Operation adds new neighbor to the neighbor set
ClearNextTasks()	Operation clears neighbor set and deletes all outgoing arcs
GetNextTasks()	Operation returns neighbor set
NextTaskContains()	Operation checks if neighbor set contains given neighbor

3.2. Transition from Primitive Structure to BPEL Process

A BPEL process is hierarchically structured activity set of sequential execution. It does not allow arbitrary cycles or goto-like constructions – the process constructions are either sequential or inclusive. It means that situation when a construction A, let us say, ‘while’, opens and then construction B (‘flow’) opens, but then A is closed and B remains open, is not valid. The opening and closing of construction blocks should follow the LIFO principle.

There are cases when vertices of primitive structure do not follow that principle – that happens when both incoming and outgoing arc count is greater than 1, i.e., the vertex both synchronizes and splits the process flow. To avoid such problems corresponding vertex is divided in two vertices – the first for synchronizing the flow but the second for splitting it. The division of vertices helps to identify patterns correctly. It also facilitates the finding of the starting and ending activities of each pattern. Figure 7 shows a fragment of workflow where a transition synchronizes the flow and immediately splits it.

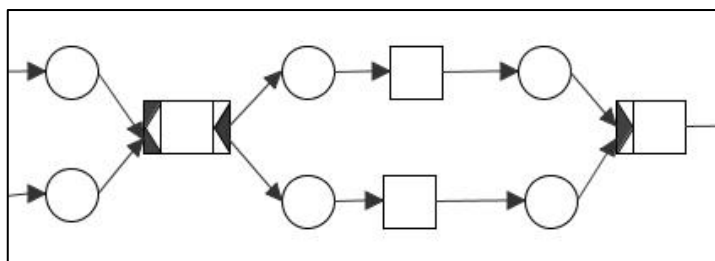


Figure 7. The synchronization and splitting of the process flow in YAWL

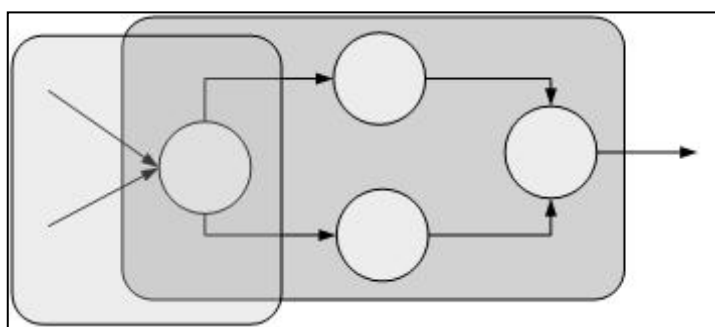


Figure 8. Pattern overlapping in primitive structure

Figure 8 shows overlapping of the patterns in primitive structure. The division is carried out by traversing all the vertices of primitive structure and dividing those whose both incoming and outgoing arc count is greater than 1. The original vertex preserves its incoming (synchronizing) arcs, while the outgoing (splitting) arcs are added to new vertex. Both vertices are connected with new arc.

If $I(A)$ – set of incoming arcs of vertex A , $O(A)$ – set of outgoing arcs of vertex A , but $|I(A)| > 1 \wedge |O(A)| > 1$, then define vertex A' :

- $O(A') = O(A)$ – copy the outgoing arcs of ‘old’ vertex to ‘new’ vertex;
- $O(A) = I(A') = \{A \Rightarrow A'\}$ – replace the outgoing arcs of ‘old’ vertex with a single arc to ‘new’ vertex, which also forms the set of incoming arcs for ‘new’ vertex.

After the primitive structure has been updated, pattern identification in it can be started. Most common patterns are shown in Table 3.

Table 3. Most common patterns

Pattern	BPEL element	Description
Simple flow	Sequence	Allows defining a set of activities that will be invoked in an ordered sequence
Loop	While, Repeat-Until	Repeating activities
Parallel flow	Flow	Allows defining a set of activities that will be invoked in parallel. The Flow is considered finished when all parallel flows within are finished.
Exclusive choice	If	Case construct for branching the flow.

The primitive structure is processed beginning with its starting activity. This activity is transformed to BPEL Receive activity, immediately followed by Reply activity. If the count of outgoing arcs of starting activity is equal to 1, the Receive-Reply pair is enclosed by Sequence Activity. If there is more than one outgoing arc, the pair is put within Pick activity. After the starting activity algorithm recursively processes its neighbors – firstly it traverses the primitive structure using depth-first search until the end activity of initially identified pattern is found, then traverses the next neighbor of initial activity using breadth-first search. When all neighbors and the activity block have been traversed and mapped onto BPEL process, algorithm continues with next block. To find the end activity of a pattern following algorithm is used – iterate through first neighbor of each activity in loop and check if its incoming arc type is equal to outgoing arc type of initial activity. To avoid mistakes when there is, for example, another If construction within initial If construction, counter is introduced – it increases with each nested construction and decreases when nested construction is closed. The end activity of the block is found when counter becomes equal to 0.

Loops within the primitive structure are identified using Tarjan SCC algorithm [11]. The result is a list of cycles, where cycles are implemented as a list of activities. Looping activities are processed like other activities – every neighbor of starting activity by depth-first search until the block end activity, then breadth-first remaining neighbors.

3.3. Requirements to the YAWL Workflow

To be able to transform the YAWL workflow successfully, the workflow must conform to some requirements. Firstly, it should not contain patterns, which have no analog constructions in BPEL, for example, the passing of process control to an activity residing outside the synchronized block, i.e. – goto-like construction. BPEL directly supports 13 patterns out of 20, discussed in [12]. Table 4 lists unsupported patterns and possible workarounds.

Secondly, the incoming and outgoing messages are associated with specific process instance using correlation sets. YAWL lacks concept of correlation sets, because each workflow instance (case) is started by its clients (users), thus creating an instance in execution environment. This environment manages the workflows and offers to users corresponding options, based on the state of instance and its specification [14]. The variable which could be used as an correlation set variable must be created in the workflow or during the transformation and finally added to each defined data type used in BPEL messages.

Table 4. Problematic patterns

Pattern	Possible workaround
Multi-merge	BPEL offers no support for Multi-merge pattern, as it does not allow for two active threads following the same path without creating new instances of another process
Discriminator	BPEL offers no direct support for Discriminator pattern; there are no structured activities which can be used for implementing it
Arbitrary cycles	Only structured loops like while and repeat-until are allowed. There are no goto-like constructs in BPEL.
Multiple instances with runtime knowledge; Multiple instances without runtime knowledge	A pick activity within a while loop is used, enabling repetitive processing, triggered by three different messages: one indicating that a new instance is required, one indicating the completion of a previously initiated instance, and one indicating that no more instances need to be created [12].
Interleaved parallel routing	Multiple scopes within a flow construct that complete for a single shared variable whose access is serialized. The order of the scopes is arbitrary but serial [13]. This solution is not applicable if one occurrence of the interleaved parallel routing pattern is embedded within another occurrence, because BPEL serializable scopes are not allowed to be nested.
Milestone	Poll within a while/repeat-until loop. Poll within a while/repeat-until loop.

Thirdly, support of human tasks – all BPEL activities related to exchange of information with process partners are perceived as web service operations, i.e., BPEL has no concept of “Human interaction”. To fill this gap several BPEL extensions are proposed, for example, BPEL4People [15] – OASIS is working on standardizing this extension, while IBM offers implementation in its WebSphere environment [16].

Last but not least, workflow definition must correctly define all the branching conditions, lest transformed BPEL process’ While, Repeat/Until and If blocks contain incorrect values.

3.4. Example

The example for simulation consists of three blocks – AND-flow, proposed ‘bottleneck’ and possible infinite loop (see Figure 9). The user of e-service is prompted to enter his/hers email address and phone number – AND-flow. Then some worker/service checks the data and prepares answer – possible ‘bottleneck’. Finally the reminder to user is sent until he/her turns up for the answer – possible infinite loop. The activity ‘Prepare answer’ is assigned to YAWL resource “User1”.

12 process instances were created – in 6 of them both e-mail address and phone number were provided, in other 6 just e-mail address. As mentioned before, CPN Tools cannot simulate OR-flows – the only way to diagnose waiting processes is using ‘Basic Log Analysis’ from ProM toolset (see Figure 10).

Next block is possible bottleneck. As Figure 11 shows, the availability of only one resource token creates a stop in other processes – 5 tokens in place “p4”. Transition “Timeout” was created to add artificial delay. Of course, it could be implemented using timed data types for all colsets used in net, but for the demonstrative purposes timed net is not used.

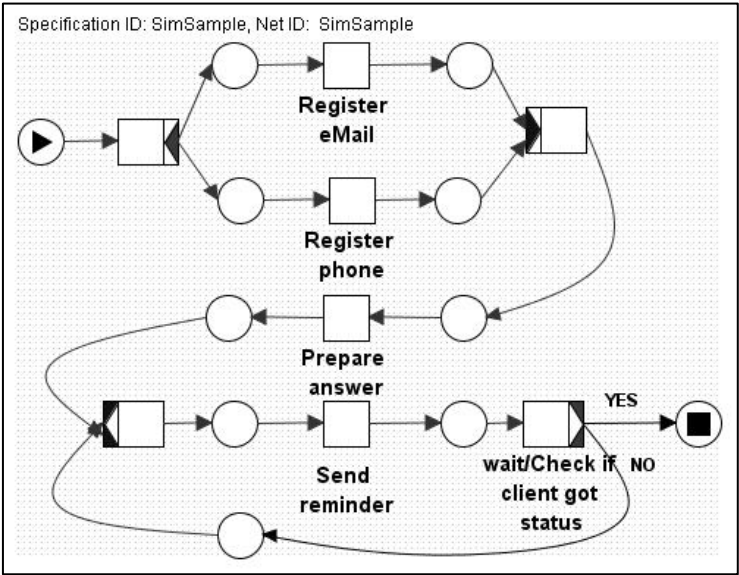


Figure 9. YAWL workflow - the simulation example

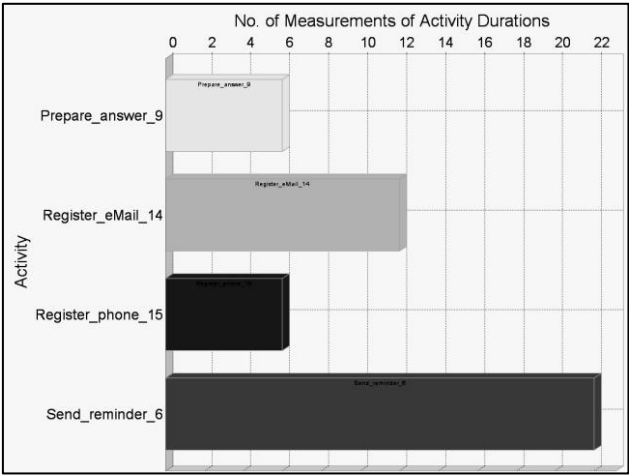


Figure 10. Basic Log Analysis – results

Figure 11 also shows 6 instances moving through loop block – possible candidate for an infinite loop.

4. Conclusion

The proposed approach of business process modeling, at first creating the model in academic language to be able to perform mathematically based analysis; and then

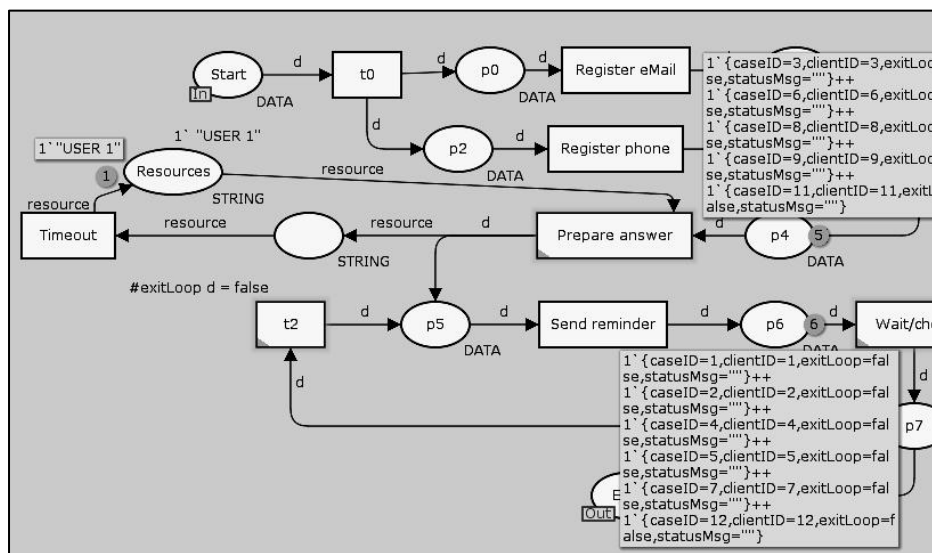


Figure 11. Bottleneck in Petri net

transforming it to the process described in business language with wider supported tool choice, is quite successful. The main benefit of this approach is creation of primitive structure and traversal and pattern identifying algorithm, which allows the transformation from YAWL workflow to any other hierarchical language, both academic and business. The traversal and identifying algorithm is applicable to the processes defined in other languages; however, these processes must be altered correspondingly to avoid the pattern overlapping problem.

The simulation model has some flaws – simpler design problems, for example, bottleneck and dead end identification can be achieved with this approach; however, more complex problems, such as deadlock identification or the operation of cancellation region, could not be resolved. To be honest, it has more to do with Petri nets, used in the simulation model, because these do not support multiple process instances or cancellation regions. It is possible that using the simulation model based on other mathematical framework, these problems could be solved.

The proposed transformation successfully recognizes the patterns used in YAWL workflow and rendered the structure of the process; however, it misses some important process description blocks. Some of them cannot be created using the proposed approach, such as faultHandlers, because simulation model used, based on Petri nets, does not support the simulation of exception handling. Some blocks simply are not processed – variables and partnerLinks fall into this category; while some are not possible to create at all, like correlationSets.

Summarizing the results, authors conclude that approach proposed in this article must be developed further. The main challenges are – the transformation must be able to create primitive structure from more complicated workflow, where patterns such as Foreach or arbitrary loops are used, at the same time not forgetting about pattern overlapping problem. Other simulation models must be examined to see if they could resolve the problems proved too challenging for model used in this article, but transformation model has to be able to create variable and partnerLinks blocks. Last but

not least, the requirements against the YAWL workflow model must be defined, so it could be possible to automatically create both `faultHandlers` and `correlationSets` blocks.

References

- [1] M. Weske "Business Process Management", Springer 2007, p. 169
- [2] W. M. P. van der Aalst, A. H. M. ter Hofstede: YAWL: Yet Another Workflow Language. *Inf. Syst.*, vol. 30(4), pp 245–275 (2005)
- [3] C. Ouyang, A.H.M. ter Hofstede, M. La Rosa, M. Rosemann, K. Shortland and D. Court, Camera, Set, Action: Automating Film Production via Business Process Management. In *Proceedings of the International Conference "Creating Value: Between Commerce and Commons"*, Brisbane, Australia, 2008
- [4] YAWL4Health, <http://www.yawlfoundation.org/casestudies/health>
- [5] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar: Workflow Control-Flow Patterns: A Revised View. *BPM Center Report BPM-06-22*, BPMcenter.org (2006)
- [6] A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C. J. Fidge: Workflow Simulation for Operational Decision Support Using Design, Historic and State Information. *Proceedings of the 6th International Conference on Business Process Management (BPM 2008)*, 2008, Milan, Italy. LNCS, vol. 5240, pp 196 – 211. Springer (2008).
- [7] W.M.P. van der Aalst, B.F. van Dongen, C.W. Gunther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A.J.M.M. Weijters. *ProM 4.0: Comprehensive Support for Real Process Analysis*. In J. Kleijn and A. Yakovlev, editors, *Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007)*, LNCS, vol. 4546, pp 484-494. Springer, Berlin (2007)
- [8] K. Jensen, L.M. Kristensen, and L. Wells: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *International Journal on Software Tools for Technology Transfer*, vol. 9(3-4), pp 213-254 (2007)
- [9] Chun Ouyang , Marlon Dumas , Wil M. P. Van Der Aalst , Arthur H. M. Ter Hofstede , Jan Mendling, From business process models to process-oriented software systems, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, v.19 n.1, p.1-37, August
- [10] J. Koehler and R. Hauser. Untangling unstructured cyclic flows - A solution based on continuations. In R. Meersman, Z. Tari, W.M.P. van der Aalst, C. Bussler, and A. Gal et al., editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004*, volume 3290 of *Lecture Notes in Computer Science*, pages 121–138, 2004.
- [11] Strong Connectivity, <http://www.ics.uci.edu/~eppstein/161/960220.html#sca>
- [12] P. Wohed, W. van der Aalst, M. Dumas, A. H. M. ter Hofstede: Pattern Based Analysis of BPEL4WS. *FIT Technical Report, FIT-TR-2002-04*, Queensland University of Technology, Brisbane, 2002
- [13] M. Havey: *Essential Business Process Modeling*, p. 141, O'Reilly (2005)
- [14] W. M. P. van der Aalst, L. Aldred, M. Dumas, T. A. H. M. Hofstede: Design and Implementation of the YAWL System. *Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAISE'04)*, Riga, Latvia, LNCS vol. 3084, pp 142–159, Springer (2004).
- [15] Holmes T., Vasko M., Dustdar S.: VieBOP: Extending BPEL Engines with BPEL4People. *16th Euromicro International Conference on Parallel, Distributed and network-based Processing 2008*, 547-555. February 2008.
- [16] WS-BPEL Extension for People, <http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/>

Grammatical Aspects: Coping with Duplication and Tangling in Language Specifications¹

Andrey BRESLAV

ITMO University, Saint-Petersburg, Russia

Abstract. For the purposes of tool development, computer languages are usually described using context-free grammars with annotations such as semantic actions or pretty-printing instructions. These specifications are processed by generators which automatically build software, e.g., parsers, pretty-printers and editing support.

In many cases the annotations make grammars unreadable, and when generating code for several tools supporting the same language, one usually needs to duplicate the grammar in order to provide different annotations for different generators.

We present an approach to describing languages which improves readability of grammars and reduces the duplication. To achieve this we use Aspect-Oriented Programming principles. This approach has been implemented in an open-source tool named GRAMMATIC. We show how it can be used to generate pretty-printers and syntax highlighters.

Keywords. DSL, Grammar, Aspect

Introduction

With the growing popularity of Domain-Specific Languages, the following types of supporting tools are created more and more frequently:

- Parsers and translators;
- Pretty-printers;
- Integrated Development Environment (IDE) add-ons for syntax highlighting, code folding and outline views.

Nowadays these types of tools are usually developed with the help of generators which accept language specifications in the form of annotated (context-free) grammars.

For example, tools such as YACC [1] and ANTLR [2] use grammars annotated with embedded semantic actions. As an illustration of this approach first consider an annotation-free grammar for arithmetic expressions (Listing 1). To generate a translator, one has to annotate the grammar rules with embedded semantic actions. Listing 2 shows the rule `expr` from Listing 1 annotated for ANTLR v3.

¹This work was partly done while the author was a visiting PhD student at University of Tartu, under a scholarship from European Regional Development Funds through Archimedes Foundation. Author would like to acknowledge the advice received from Professor Marlon Dumas.

```

expr : term ((PLUS | MINUS) term)* ;
term : factor ((MULT | DIV) factor)* ;
factor : INT | '(' expr ')' ;

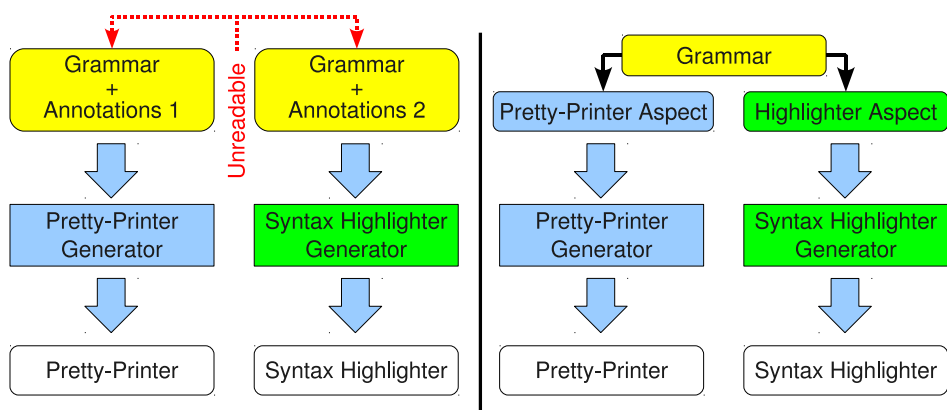
```

Listing 1. Grammar for arithmetic expressions

```

expr returns [int result] :
  t=term {result = t;}
  ((int sign = 1;) (PLUS | MINUS {sign = -1;})
    t=term {result += sign * t;});

```

Listing 2. Annotated grammar rule**Figure 1.** Generating two supporting tools for the same language

As can be seen, the context-free grammar rule is not easily readable in Listing 2 because of the actions' code interfering with the grammar notation. This problem is common for annotated grammars. We will refer to it as *tangled grammars*.

In most applications we need to create several supporting tools for the same language (see Figure 1, left side). In such a case one uses different generators to obtain different programs (e.g., PRETZEL [3] to build a pretty-printer and xTEXT [4] to create an Eclipse editor). Each generator requires its own specific set of annotations, and the developer has to write the same grammar several times with different annotations for each generator. Besides the duplication of effort, when the language evolves, this may lead to inconsistent changes in different copies of the grammar, which may cause issues which are hard to detect. We will refer to this problem as *grammar duplication*.

This paper aims at reducing tangling and duplication in annotated grammars. A high-level view of our approach is illustrated in Figure 1 (right side): the main idea is to separate the annotations from the grammar by employing the principles similar to those behind the AspectJ language [5], this leads to a notion of a *grammatical aspect*. Our approach is implemented in an open-source tool named GRAMMATIC².

²The tool is available at <http://grammatic.googlecode.com>

In Section 1 we briefly describe the main notions of aspect-oriented programming in AspectJ. Definitions related to grammatical aspects and their formal semantics are given in Section 2. Section 5 studies the applications of GRAMMATIC to generating syntax highlighters and pretty-printers on the basis of a common grammar. We analyze these applications and evaluate our approach in Section 6. Related work is described in Section 7. Section 8 summarises the contribution of the paper and introduces possible directions of the future work.

1. Background

Aspect-Oriented Programming (AOP) is a body of techniques aimed at increasing modularity in general-purpose programming languages by separating cross-cutting concerns. Our approach is inspired by AspectJ [5], an aspect-oriented extension of Java.

AspectJ allows a developer to extract the functionality that is scattered across different classes into modules called *aspects*. At compilation- or run-time this functionality is *weaved* back into the system. The places where code can be added are called *join points*. Typical examples of join points are a method entry point, an assignment to a field, a method call.

AspectJ uses *pointcuts* — special constructs that describe collections of join points to weave the same piece of code into many places. Pointcuts describe method and field signatures using patterns for names and types. For example, the following pointcut captures *calls of all public get-methods in the subclasses of the class Example which return int and have no arguments*:

```
pointcut getter() : call(public int Example+.get*())
```

The code snippets attached to a pointcut are called *advice*; they are weaved into every join point that matches the pointcut. For instance, the following advice writes a log record after every join point matched by the pointcut above:

```
after() : getter() {
    Log.write("A get method called");
}
```

In this example the pointcut is designated by its name, `getter`, that follows the keyword `after` which denotes the position for the code to be weaved into. An *aspect* is basically a unit comprising of a number of such pointcut-advice pairs.

2. Overview of the approach

Our approach employs the principles of AOP in order to tackle the problems of tangling and duplication in annotated grammars. We will use the grammar from Listing 1 and the annotated rule from Listing 2 to illustrate how the terms such as “pointcut” and “advice” are embodied for annotated grammars.

2.1. Grammatical join points

Figure 2 shows a structured representation (a syntax diagram) of the annotated rule from Listing 2. It shows the annotations attached to a symbol definition `expr`, three sym-

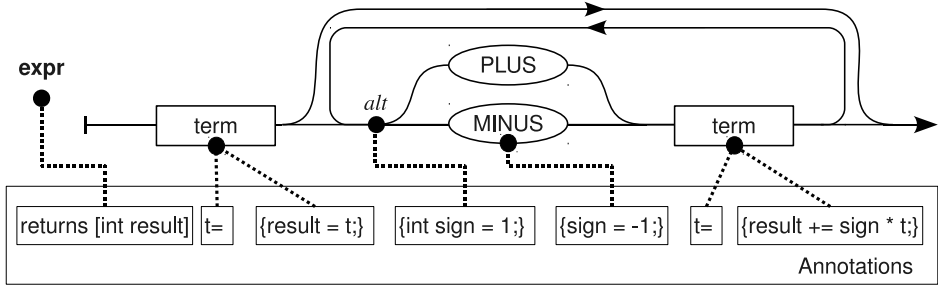


Figure 2. Annotations attached to a grammar rule

bol references: `term` (two times) and `MINUS`, and an alternative (`PLUS` | `MINUS`) (marked “*alt*” in the figure). All these are examples of *grammatical join points* (in Figure 2 they are marked with black circles).

Below we use ANTLR’s notation for context-free grammars and reflect on ASTs of this notation. To avoid confusion with ASTs of languages defined by the grammar, we will refer to the AST of the grammar itself as *grammar tree* (GT). The language of GTs is given by the following type definitions:

```

data Grammar = Grammar {rules :: Set(Rule)}
data Rule = Rule {name :: String, body :: Expression}
data Expr -- Expressions
    = SymRef {name :: String} -- Symbol reference
    | Literal {value :: String} | Empty
    | Seq {exprs :: List(Expr)} -- Sequence
    | Alt {exprs :: Set(Expr)} -- Alternative
    | Iter {expr :: Expr, multiplicity :: Mult} -- Iteration
data Mult = Star | Plus | Option
  
```

Every node in a GT can be a join point, thus we have as many join point types as data constructors in the GT language above.

2.2. Grammatical pointcuts

Pointcuts are expressions denoting sets of join points; they are described using *patterns* over the GT language. The language of pointcuts is obtained from the GT language by adding special constructs, namely wildcards and variable bindings (see Table 1). Consider some examples of patterns for the rules from Listing 1:

Table 1. Notions introduced in the pointcut language (in addition to the GT nodes)

Concrete Syntax	Abstract Syntax	Meaning
	?Type	Any instance of type Type
?	?SymRef	Any symbol
?lex	?Literal	Any lexical literal
[?]	[?]	Any sequence
{?}	{?}	Any set of alternatives
... \$v=e1 ...	$\lambda v \approx e_1 . e_2$	Variable binding

- (a) $\text{expr} : [?]$ — a rule defining a symbol “ expr ” with an arbitrary expression on the right-hand side (in Listing 1 it matches only the rule for expr);
- (b) $? : \text{term} [?]$ — a rule for any symbol, starting with a reference to a symbol named “ term ” (also matches only the rule for expr);
- (c) $? : ? [?]^*$ — a symbol reference followed by a star iterating an arbitrary sequence (matches the rules for expr and term).

Note that constructors of the GT language are used in the pattern language abstract syntax; their typing is relaxed in order to accommodate wildcards written as $?Type$. The example (b) from above can be written in the abstract syntax (“ $:$ ” denotes the *cons* operation on lists) as follows:

$Rule(?SymRef, Seq(SymRef(\text{term}) : ?List(Expr)))$.

Variables are used to label subexpressions and refer to them later. Variable definitions and usages are prefixed with the dollar sign. Consider the following example:

$? : \$tr = ? ([?] \$tr)^*$

Here the variable tr is defined with the pattern $?$ (any symbol) which means that all usages of the variable will match only occurrences of the same symbol. This example matches the rule for expr because the same symbol term is referenced in the positions matched by the variable tr . In abstract syntax variables are defined with λ -abstraction:

$\lambda tr \approx ?SymRef . Seq(tr : Iter(Seq(?List(Expr) : tr), Star))$

In this paper we assume that GT nodes are *not* shared, and subtrees of the same shape occurring at different positions are distinct. This corresponds to having unique numbers implicitly assigned to nodes, below we refer to these numbers as *identity labeling* and write them in superscripts: for example. $Rule^0(SymRef^1(x), SymRef^2(y))$. In this regard, note that a variable is in general bound to a *set* of GT nodes: if we match the rule for expr against the pattern in the example above, the variable tr will be bound to a set comprised by two distinct references to term , with distinct identity labels.

2.3. Grammatical advice

As mentioned above, the main idea of our approach is to separate annotations used by various generators from grammars. Treated in this manner, annotations become *grammatical advice* (by analogy with AspectJ’s advice). In general, a generator may need a very rich annotation system, and grammatical advice must be capable of expressing it. In our approach, we use a *generic advice language*, which represents the annotations as sets of name-value pairs which we call *attributes*. Examples of such pairs are given in Table 2 which shows all the predefined value types. Developers can introduce user-defined types, but we do not describe this mechanism here, in order to save space. The annotations in Figure 2 may be represented, for example, as values of type String (other representations are also possible).

As the usage of the term “attribute” may be misleading in this context, we would like to note that the approach presented here does not directly correspond to attribute grammars [6]. In fact, grammars with annotations do not have any particular execution semantics (each generator interprets the annotations in its own way), as opposed to attribute grammars which have a fixed execution semantics. One can describe attribute grammars using our approach and define corresponding semantics in a generator, but this is just an example application.

Table 2. Predefined value types

Example	Value type
<code>int = 10</code>	Integer
<code>str = 'Hello'</code>	String
<code>var = SomeName</code>	Name literal
<code>rec = {b = c; d = 5}</code>	Annotation
<code>seq = [1 a b 'str']</code>	Sequence of values

2.4. Grammatical aspects

Now, having described all the components, we can assemble a *grammatical aspect* as a set of pointcut-advice pairs. Usage of grammatical aspects is illustrated by Figure 1.

An aspect consists of an optional *grammar annotation* and zero or more *annotation rules*. Annotation rules associate grammatical pointcuts (rule patterns) with advice (annotations). Here is an example of an annotation rule:

```
expr : $tr=? ([?] $tr)*    // pointcut (pattern)
      $tr {varName = 't'} ; // advice (annotation)
```

In a simple case exemplified here, an annotation `{varName = 't'}` is attached to GT nodes to which a variable `tr` is bound. For more complicated cases, one can define arbitrarily nested *subpatterns* — patterns being matched against nodes situated in a particular GT subtree. Subpatterns are marked with “@” symbol. For example, the following construct attaches an attribute named `varName` to each reference to the symbol `term` inside a rule matched by a top-level pattern:

```
expr : [?]                // pointcut (pattern)
      @$tr=(term):        // pointcut (subpattern)
      $tr {varName = 't'} ; // advice (annotation)
```

This example illustrates the typical usage of subpatterns where all annotations are associated with a variable bound to the whole pattern. As a shorthand for this situation the variable can be omitted (it will be created implicitly). Using this shorthand we can abridge the previous example to the following:

```
expr : [?]
      @term: { varName = 't' } ;
```

3. Semantics of pointcuts and aspects

To precisely define what happens when an aspect is applied to a language specification, in this section we outline formal semantics of the constructs described above. The key point in this formalization is to characterize pointcut matching by establishing a system of inequalities over sets of terms³ corresponding to pointcuts. Solutions of these inequalities give valuations to variables defined inside patterns and subpatterns.

Definition 1 (System of set inequalities). *Let \mathcal{V} be an infinite set of variables, \mathcal{T} — a finite set of types, \mathcal{C} — a finite set of value constructors, and $\bar{\mathcal{C}}$ — the set of all possible identity-labeled terms that use constructors from \mathcal{C} and variables from \mathcal{V} . A system of set inequalities is a finite set of expressions of the following forms:*

³Since terms are trees, in this section we use “GT node” and “term” interchangeably.

- $S_1 \subseteq S_2$, where $S_i \subseteq \bar{\mathcal{C}}$,
- or $\forall x \in S.P(x)$, where $x \in \mathcal{V}$.

Given such a system S , a minimal function $\sigma : \mathcal{V} \rightarrow 2^{\bar{\mathcal{C}}}$ is called a solution of S , iff replacing every occurrence of every $v \in \mathcal{V}$ in S by $\sigma(v)$ turns all the entries of S into true statements.

In addition to the common set operations, we use the following conventions:

- $Strip(t)$ denotes a term t with its identity labels removed.
- $\llbracket T \rrbracket$, where $T \in \mathcal{T}$ denotes a set of all values of type T .
- If $C \in \mathcal{C}$, then $C(S_1, \dots, S_n) = \{C(s_1, \dots, s_n) \mid s_1 \in S_1, \dots, s_n \in S_n\}$.
- For $t \in \bar{\mathcal{C}}$, $Subterms(t)$ is the set of all direct and indirect subterms of t .
- $L_1 \oplus L_2 = \{l_1 \oplus l_2 \mid l_1 \in L_1, l_2 \in L_2\}$, where \oplus denotes list concatenation.
- A single value can be treated as a singleton set or singleton list containing this value, if required by the context.

Definition 2. (Pattern projection) Let P be a pattern and \mathcal{V}_P be a set of all variables defined inside this pattern, then for $v \in \mathcal{V}_P$ the pattern P_v is called a v -projection of P if it is acquired from P by substituting all the variables except v by their corresponding patterns defined in λ -abstractions.

For example, consider the following pattern: $\$a=? : \$b=\$a$. In abstract syntax it is written as follows: $P = \lambda a \approx ?SymRef . \lambda b \approx a . Rule(a, b)$. Projections for P are $P_a = \lambda a \approx ?SymRef . Rule(a, a)$ and $P_b = \lambda b \approx ?SymRef . Rule(?SymRef, b)$.

Figure 3 defines a function \mathcal{S} that takes a variable and a pattern as arguments and returns a system of set inequalities. We denote variables v which are supposed to be fresh with respect to the rest of the system as v^\dagger . For example, usage of this notation in the first rule in Figure 3 means that fresh variables v_i^\dagger are introduced every time this rule is applied.

Definition 3 (Matching). Given a pointcut expression P , a GT t and $R \in \mathcal{V} \setminus \mathcal{V}_P$, let

$$S_P^R(t) = \{R = t\} \cup \bigcup_{v \in \mathcal{V}_P} \mathcal{S}(R, P_v)$$

Then, P matches t iff there is a solution σ to $S_P^R(t)$.

To incorporate a subpattern P_1 into this definition, it is enough to consider an extended system $S_{P, P_1}^R(t) = S_P^R(t) \cup S_{P_1}^{R_1}(t) \cup \{R_1 \subseteq Subterms(R)\}$. Arbitrarily nesting subpatterns are easily expressible in this manner.

To save space, let us denote $SymRef^n(X)$ as X^n . For the example pattern P given above and a term t : $x : x$, or, in abstract syntax: $t = Rule^0(SymRef^1(X), SymRef^2(X)) = Rule^0(X^1, X^2)$ the system $S_P^R(t)$ is comprised of the following inequalities:

$$\begin{aligned}
\mathcal{S}(V, \mathbf{Type}(t_1, \dots, t_n)) &= \{V \subseteq \mathbf{Type}(v_1^\dagger, \dots, v_n^\dagger)\} \cup \bigcup_1^n \mathcal{S}(v_i^\dagger, t_i), \\
&\text{where } \mathbf{Type} \subseteq \{\text{Grammar}, \text{Rule}, \text{Iter}, \text{SymRef}, \text{Literal}, \text{Empty}\} \\
\mathcal{S}(V, ?\text{Type}) &= \{V \subseteq \llbracket \text{Type} \rrbracket\} \\
\mathcal{S}(V, \text{Seq}(e_1, \dots, e_n)) &= \{V \subseteq \text{Seq}(L^\dagger)\} \cup \{L^\dagger = L_1^\dagger \oplus \dots \oplus L_n^\dagger\} \cup \bigcup_1^n \mathcal{S}(L_i^\dagger, e_i) \\
\mathcal{S}(V, [?]) &= \{V \subseteq \llbracket \text{List}(\text{Expr}) \rrbracket\} \\
\mathcal{S}(V, \text{Alt}(e_1, \dots, e_n, \{?\})) &= \{V \subseteq \text{Alt}(S^\dagger)\} \cup \bigcup_1^n \mathcal{S}(t_i^\dagger, e_i) \cup \bigcup_1^n \{t_i^\dagger \subseteq S^\dagger\} \\
\mathcal{S}(V, \lambda v \approx e_1 . e_2) &= \mathcal{S}(v, e_1) \cup \mathcal{S}(V, e_2) \cup \{\text{varax}(v)\}, \\
&\text{where } \text{varax}(v) = \forall x', x'' \in v. \text{Strip}(x') = \text{Strip}(x'') \\
\mathcal{S}(V, v) &= \{V \subseteq v\}
\end{aligned}$$

Figure 3. Transforming a pattern into a system of inequalities

$$\begin{array}{lll}
\mathcal{S}(R, P_a) : & \mathcal{S}(R, P_b) : & R = \text{Rule}^0(X^1, X^2) \\
a \subseteq \llbracket \text{SymRef} \rrbracket & b \subseteq \llbracket \text{SymRef} \rrbracket & \\
\text{varax}(a) & \text{varax}(b) & \\
R \subseteq \text{Rule}(v_1, v_2) & R \subseteq \text{Rule}(v_3, v_4) & \\
v_1 \subseteq a & v_3 \subseteq \llbracket \text{SymRef} \rrbracket & \\
v_2 \subseteq a & v_4 \subseteq b &
\end{array}$$

Recall that $\text{varax}(v)$, as defined in Figure 3, is a statement saying that all terms in the set denoted by v are equal modulo the identity labeling. If we change t to be $t' = \text{Rule}^0(X^1, Y^2)$, then $\text{varax}(a)$ will not hold, and the system will be unsatisfiable. But with the given t this system has a solution: $\sigma(a) = \{X^1, X^2\}$, $\sigma(b) = \{X^2\}$. This solution reflects the fact that b is only bound to the second occurrence of X , whereas a is bound to both occurrences.

To complete the semantics of aspects, note that an aspect rule r gives a function $A_r(v)$ returning an annotation to be associated to every $n \in \sigma(v)$. To apply a particular rule r (a pointcut P_r and the annotation function A_r), for every node t in the GT, if there is a solution σ for $S_{P_r}^R(t)$, then for every $v \in \mathcal{V}_{P_r}$ associate $A_r(v)$ with every $n \in \sigma(v)$. If there are two annotations associated to the same node, they are merged (duplicate attributes, if any, are kept). Naturally, the grammar annotation (if present) is simply associated to the whole grammar.

4. Implementation

Our approach is implemented as an open-source tool GRAMMATIC written in Java. The tool provides concrete syntax for grammar specifications and aspects and implements the matching and application semantics given above.

GRAMMATIC works as a front-end to generators that use its API. First, it takes a specification and applies aspects. If the application was successful, the resulting structure (GT nodes with attached annotations) is passed to the generator which processes it as a whole and needs no information about aspects. To use a pre-existing tool, for example, ANTLR, with grammatical aspects, one can employ a small generator which calls GRAMMATIC to apply aspects to grammars, and produces annotated grammars in the ANTLR format.

From the practical point of view, it is important that the definitions in the previous section allow a single join point to match more than once. Depending on the grammar,

this behavior may be intended by the developer, or not. To provide some means of control over this behavior, patterns and subpatterns may be preceded by a *multiplicity directive*. For example:

```
(0..1) ? : $tr=? ([?] $tr)* // pointcut with multiplicity
    // some advice
```

A multiplicity directive determines a number of times the pattern is allowed to match. The default multiplicity is $(1..*)$ which means that each pattern with no explicit multiplicity is allowed to match one or more times. When an aspect is applied to a grammar, if the actual number of matches goes beyond the range allowed by a multiplicity directive, an error message will be generated. In the example above, there will be an error when matching against the grammar from Listing 1 because the pattern matches two rules: `expr` and `term`, which violates the specified multiplicity $(0..1)$.

5. Applications

In this section we show how one can make use of grammatical aspects when generating syntax highlighters and pretty-printers on the basis of the same grammar.

5.1. Specifying syntax highlighters

A syntax highlighter generator creates a highlighting add-on for an IDE, such as a script for `vim` editor or a plug-in for Eclipse. For all targets the same specification language is used: we annotate a grammar with *highlighting groups* which are assigned to occurrences of terminals. Each group may have its own color attributes when displayed. Common examples of highlighting groups are *keyword*, *number*, *punctuation*.

In many cases syntax highlighters use only lexical analysis, but it is also possible to employ light-weight parsers [7]. In such a case grammatical information is essential for a definition of the highlighter. Below we develop an aspect for the Java grammar which defines groups for keywords and for *declaring occurrences* of class names and type parameters. A declaring occurrence is the first occurrence of a name in the program; all the following occurrences of that name are *references*. Consider the following example:

```
class Example<A, B extends A> implements Some<? super B>
```

This illustrates how the generated syntax highlighter should work: the declaring occurrences are underlined (occurrences of `?` are always declaring) and the keywords are shown in bold. This kind of highlighting is helpful especially while developing complicated generic signatures.

Listing 3 shows a fragment of the Java grammar [8] which describes class declarations and type parameters. In Listing 4 we provide a grammatical aspect which defines three highlighting groups: *keyword*, *classDeclaration* and *typeParameterDeclaration*, for join points inside these rules.

Each annotation rule from Listing 4 contains two subpatterns. The first one is `?lex`: it matches every lexical literal. For example, for the first rule it matches `'class'`, `'extends'` and `'implements'`; the highlighting group *keyword* is assigned to all these literals.

The second subpattern in each annotation rule is used to set a corresponding highlighting group for a declaring occurrence: for classes and type parameters it matches `IDENTIFIER` and for wildcards — the `'?'` literal.

```

normalClassDeclaration
    : 'class' IDENTIFIER typeParameters?
      ('extends' type)? ('implements' typeList)? classBody ;
classBody
    : '{' classBodyDeclaration* '}' ;
typeParameters
    : '<' typeParameter (',' typeParameter)* '>' ;
typeParameter
    : IDENTIFIER ('extends' bound)? ;
bound
    : type ('&' type)* ;
type
    : IDENTIFIER typeArgs? ('.' IDENTIFIER typeArgs?)* ('[' ']'')*
    : basicType ;
typeArgs
    : '<' typeArgument (',' typeArgument)* '>' ;
typeArgument
    : type
    : '?' (('extends' | 'super') type)? ;

```

Listing 3. Class declaration syntax in Java 5

```

? : 'class' IDENTIFIER [?]
  @?lex: { group = keyword } ;
  @IDENTIFIER: { group = classDeclaration } ;
typeParameter : IDENTIFIER [?]
  @?lex: { group = keyword } ;
  @IDENTIFIER: { group = typeParameterDeclaration } ;
typeArgument : [?]
  @?lex: { group = keyword } ;
  @'?': { group = typeParameterDeclaration } ;

```

Listing 4. Highlighting aspect for class declarations in Java

When the aspect is applied to the grammar, GRAMMATIC attaches the *group* attribute to the GT nodes matched by the patterns in the aspect. The obtained annotated grammar is processed by a generator which produces code for a highlighter.

5.2. Specifying pretty-printers

By applying a different aspect to the same grammar (Listing 3), one can specify a pretty-printer for Java. A pretty-printer generator relies on annotations describing how tokens should be aligned by inserting whitespace between them.

In Listing 5 these annotations are given in the form of attributes *before* and *after*, which specify whitespace to be inserted into corresponding positions. Values of the attributes are *sequences* ([...]) of strings and name literals *increaseIndent* and *decreaseIndent* which control the current level of indentation.

The most widely used values of *before* and *after* are specified in a *grammar annotation* by attributes *defaultBefore* and *defaultAfter* respectively, and not specified for each token individually. In Listing 5 the default formatting puts nothing

```

{ // Grammar annotation
  defaultAfter = [ ' ' ];
  defaultBefore = [ ' ' ];
}

classBody : '{' classBodyDeclaration* '}'
  @'{' : { after = [ '\n' increaseIndent ] } ;
  @classBodyDeclaration : { after = [ '\n' ] } ;
  @'}' : {
    before = [ decreaseIndent '\n' ];
    after = [ '\n' ];
  };
typeParameters : '<' typeParameter (',' typeParameter)* '>'
  @'<' : { after = [ ' ' ] } ;
  @typeParameter : { after = [ ' ' ] } ;

```

Listing 5. Pretty-printing aspect for class declarations in Java

before each token and a space — after each token; it applies whenever no value was set explicitly.

6. Discussion

This paper aims at coping with two problems: tangled grammars and grammar duplication. When using GRAMMATIC, a single annotated grammar is replaced by a *pure* context-free grammar and a set of grammatical aspects. This means that the problem of *tangled grammars* is successfully addressed.

This also means that the grammar is written down only once even when several aspects are applied (see the previous section). But if we look at the aspects, we see that the patterns carry on some extracts from the grammar thus it is not so obvious whether our approach helps against the problem of *duplication* or not. Let us examine this in more details using the examples from the previous section.

From the perspective of grammar duplication, the worst case is an aspect where all the patterns are exact citations from the grammar (no wildcards are used, see Listing 5). This means that a large part of the grammar is completely duplicated by those patterns. But if we compare this with the case of conventional annotated grammars, there still is at least one advantage of using GRAMMATIC. Consider the scenario when the grammar has to be changed. In case of conventional annotated grammars, the same changes must be performed once for each instance of the grammar and there is a risk of inconsistent changes which are not reported to the user. In GRAMMATIC, on the other hand, a developer can control this using *multiplicities*: for example, check if the patterns do not match anything in the grammar and report it (since the default multiplicities require each pattern to match at least once, this will be done automatically). Thus, even in the worst case, grammatical aspects make development less error-prone.

Using wildcards and subpatterns as it is done in Listing 4 (i) reduces the duplication and (ii) makes a good chance that the patterns will not need to be changed when the grammar changes. For example, consider the first annotation rule from Listing 4: this rule works properly for both Java version 1.4 and version 5 (see List-


```

classDeclaration
: 'class' IDENTIFIER ('extends' type)?
  ('implements' typeList)? classBody ;

```

Listing 6. Class declaration rule in Java 1.4

ing 6 and Listing 3 respectively). The pointcut used in this rule is sustainable against renaming the symbol on the left-hand side (`classDeclaration` was renamed to `normalClassDeclaration`) and structural changes to the right-hand side (type parameters were introduced in Java 5). The only requirement is that the definition should start with the `'class'` keyword followed by the `IDENTIFIER`.

In AOP, the duplication of effort needed to modify pointcuts when the main program changes is referred to as the *fragile pointcut problem* [9]. Wildcards and subpatterns make pointcuts more *abstract*, in other words, they widen the range of join points matched by the pointcuts. From this point of view, wildcards help to abstract over the contents of the rule, and subpatterns — over the positions of particular elements within the rule. The more abstract a pointcut is, the less duplication it presents and the less fragile it is.

The most abstract pointcut does not introduce any duplication and is not fragile at all. Unfortunately, it is also of no use, since it matches any possible join point. This means that eliminating the duplication completely from patterns is not technically possible. Fortunately, we do not want this: if no information about a grammar is present in an aspect, this makes it much less readable because the reader has no clue about how the annotations are connected to the grammar. Thus, there is a trade-off between the readability and duplication in grammatical aspects and a developer should keep pointcuts as abstract as it is possible without damaging readability.

To summarize, our approach allows one to keep a context-free grammar completely clean by moving annotations to aspects and to avoid any unnecessary duplication by using abstract pointcuts.

7. Related work

Several attribute grammar (AG) systems, namely JASTADD [10], SILVER [11] and LISA [12], successfully use aspects to attach attribute evaluation productions to context-free grammar rules. AGs are a generic language for specifying computations on ASTs. They are well-suited for tasks such as specifying translators in general, which require a lot of expressive power. But the existence of more problem-oriented tools such as PRETZEL [3] suggests that the generic formalism of AGs may not be the perfect tool for problems like generating pretty-printers. In fact, to specify a pretty-printer with AGs one has to produce a lot of boilerplate code for converting an AST into a string in concrete syntax. As we have shown in Section 5, GRAMMATIC facilitates creation of such problem-oriented tools providing the syntactical means (grammatical aspects) to avoid tangled grammars and unnecessary duplication.

The MPS [13] project (which lies outside the domain of textual languages since the editors in MPS work directly on ASTs) implements the approach which is very close to ours. It uses aspects attached to the *concept language* (which describes abstract syntax

of MPS languages) to provide input data to generators. The implementation of aspects in MPS is very different from the one in GRAMMATIC: it does not use pointcuts and performs all the checking while the aspects are created.

There is another approach to the problems we address: parser generators such as SABLECC [14] and ANTLR [2] can work on annotation-free grammars and produce parsers that build ASTs automatically. In this way the problems induced by using annotations are avoided. The disadvantage of this approach is that the ASTs must be processed manually in a general-purpose programming language, which makes the development process less formal and thus more error-prone.

8. Conclusion

Annotated grammars are widely used to specify inputs for various generators which produce language support tools. In this paper we have addressed the problems of tangling and duplication in annotated grammars. Both problems affect maintainability of the grammars: tangled grammars take more effort to understand, and duplication, besides the need to make every change twice as the language evolves, may lead to inconsistent changes in different copies of the same grammar. We have introduced *grammatical aspects*, and showed how they may be used to cope with these problems by separating context-free grammars from annotations.

The primary contributions of this paper are:

- Formal description of grammatical aspects and their semantics.
- A tool named GRAMMATIC which provides languages for specifying grammatical pointcuts, advice and aspects, and implements the semantics of matching and aspect application.

We have demonstrated how GRAMMATIC may be used to generate a syntax highlighter and a pretty-printer by applying two different aspects to the same grammar. We have shown that grammatical aspects help to “untangle” grammars from annotations, and eliminate the unnecessary duplication. The possible negative impact of remaining duplication (necessary to keep the aspects readable) can be addressed in two ways: (i) *abstract patterns* reduce the amount of changes in aspects per change in the grammar, and (ii) *multiplicities* help to detect inconsistencies at generation time.

One possible way to continue this work is to support grammar adaptation techniques [15] in GRAMMATIC to facilitate rephrasing of syntax definitions (e.g., left factoring or encoding priorities of binary operations) to satisfy requirements of particular parsing algorithms.

Another possible direction is to generalize the presented approach to support not only grammars, but also other types of declarative languages used as inputs for generators, such as UML or XSD.

References

- [1] Stephen C. Johnson. Yacc: Yet another compiler-compiler. Technical report, Bell Laboratories, 1979.
- [2] Terence Parr. *The Definitive ANTLR Reference: Building Domain-Specific Languages*. Pragmatic Programmers. Pragmatic Bookshelf, first edition, May 2007.

- [3] Felix Gärtner. The PretzelBook. Available online at <http://www.informatik.tu-darmstadt.de/BS/Gaertner/pretzel/> (last accessed on April 28, 2010), 1998.
- [4] The Eclipse Foundation. Xtext. <http://www.eclipse.org/Xtext>, 2009.
- [5] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. An overview of AspectJ. In *ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming*, pages 327–353, London, UK, 2001. Springer-Verlag.
- [6] Donald E. Knuth. Semantics of context-free languages. *Theory of Computing Systems*, 2(2):127–145, June 1968.
- [7] Leon Moonen. Generating robust parsers using island grammars. In *WCRE '01: Proceedings of the Eighth Working Conference on Reverse Engineering (WCRE'01)*, page 13, Washington, DC, USA, 2001. IEEE Computer Society.
- [8] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification, Third Edition*. Addison-Wesley Longman, Amsterdam, 3 edition, June 2005.
- [9] Maximilian Störzer and Christian Koppen. PCDiff: Attacking the fragile pointcut problem. In *European Interactive Workshop on Aspects in Software*, Berlin, Germany, September 2004.
- [10] Görel Hedin and Eva Magnusson. JastAdd: an aspect-oriented compiler construction system. *Science of Computer Programming*, 47(1):37–58, 2003.
- [11] Eric Van Wyk, Derek Bodin, Jimin Gao, and Lijesh Krishnan. Silver: an extensible attribute grammar system. *ENTCS*, 203(2):103–116, 2008.
- [12] Damijan Rebernak and Marjan Mernik. A tool for compiler construction based on aspect-oriented specifications. In *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference*, pages 11–16, Washington, DC, USA, 2007. IEEE Computer Society.
- [13] JetBrains. Meta Programming System (MPS). <http://www.jetbrains.com/mps>, 2009.
- [14] Etienne M. Gagnon and Laurie J. Hendren. SableCC, an object-oriented compiler framework. In *TOOLS '98: Proceedings of the Technology of Object-Oriented Languages and Systems*, page 140, Washington, DC, USA, 1998. IEEE Computer Society.
- [15] Ralf Lämmel. Grammar adaptation. In José Nuno Oliveira and Pamela Zave, editors, *FME*, volume 2021 of *Lecture Notes in Computer Science*, pages 550–570. Springer, 2001.

Information Systems Integration

This page intentionally left blank

XML-based Specification of the Project Management Domain and Its Application

Solvita BĒRZIŠA¹

*Faculty of Computer Science and Information Technology,
Riga Technical University, Latvia*

Abstract. Project management is a complex process regulated by project management methodologies, standards and other requirements. Different project management information systems are used to support this process. To ensure that the project management information system delivers expected results, it should be configured according requirements of the chosen methodology. In this configuration process XML is used for standardize definition and description of project management requirements. This paper describes structure and application of the aforementioned XML schema, which is referred as XML schema for Configuration of Project Management information systems (XCPM). XCPM is based on a comprehensive project management concept model. Each entity from the concept model can be described using the XCPM schema. The paper also surveys approaches for project management information system configuration. Configuration of the change control process in the project management information system is described to illustrate application of the XCPM.

Keywords. XCPM, project management, project management information systems, PMIS, PMIS configuration, XML schema

Introduction

Project management information systems (PMIS) are one of the key elements for project management (PM) and project success [1]. However, similarly as for other types of information systems, application of PMIS would be successful if it is implemented according to requirements of a particular project management situation or project management methodology (e.g. PMBOK [2], PRINCE2 [3], RUP[4] and MSF[5]). Ideally, PMIS would adhere to all requirements of PM methodology used for a particular project. But in reality multiple projects are governed by different methodologies and regulatory requirements [6] what makes PMIS configuration a complex task. Therefore, an approach for implementation and configuration of PMIS was developed [7]. This approach uses principles of model driven configuration of packaged applications. PM methodologies and other project requirements are specified according to a standardized definition, which is subsequently used to configure a packaged project management application by means of package specific transformations. The standardized definition can be represented as a PM domain definition XML schema. This schema must be sufficiently comprehensive so that different PM methodologies can be described in terms of this definition.

¹ Corresponding Author: Solvita Bērziša, Riga Technical University, Kalku 1, Riga, Latvia; E-mail: berzisa@gmail.com

An objective of this paper is to elaborate the XML schema for specification of the PM domain. This schema can be used for configuration of PMIS and is referred as to XML for Configuration of Project Management information systems (XCPM). It is developed on the basis of the PM concept model (PMCM) that has been obtained after performing conceptual modelling of the PM domain [8]. The XCPM schema is established using information from the PMCM and by applying the concept model to XML schema transformation rules. Each entity of the PMCM model is represented in the XCPM schema. The paper describes the PMIS configuration approach, the PMCM, XCPM schema elaboration rules and structure. A PMIS configuration prototype is designed to illustrate application of the XCPM schema.

The contribution of this research is development of the XCPM schema for specification of the PM domain. The new XML schema is developed because existing PM schemas (PMXML [9] and Microsoft Project XML schema [10]) focus on operational data such as tasks, resource, and assignment rather than on specification of structural properties. The XCPM schema also includes descriptive information about PM and PM processes. Description and storage of specific PM information in the schema is allowed by abstract elements. The main distinctive feature of the proposed schema is joint representation of data, processes and knowledge.

The paper is structured as follows. Section 1 describes the PMIS configuration approach and reviews related approaches for configuration of enterprise systems and specification of its domain. Section 2 gives overview of the PMCM and its generation process. Section 3 briefly describes structure of XCPM schema and more detail description of process and knowledge area definition. Prototype of PMIS configuration is described in Section 4. Section 5 gives conclusions about XCPM application and discussions about features of future work.

1. PMIS Configuration

XML-based specification of PM domain or XCPM is a part of research on development of the model and template driven approach for configuration of PMIS according to requirements of PM methodology and other regulations [7].

1.1. Overview of Approach

The PMIS configuration and implementation process is shown in Figure 1. It starts with definition of requirements from the PM methodology or regulatory guidelines, PM process and other PM relevant information. These requirements are necessary to represent in a standardized manner. The PM requirements transformation process is organized using the graphical interface and is supported by appropriate knowledge from the PM knowledge repository. The result of this transformation is formalized specification of PM requirements in the XML document format. The formalized specification of PM requirements can be automatically transformed to configure PMIS by using PM application specific transformation scripts. The standardized representation enables using the same transformation scripts for different PM methodologies. Some specific information from the PM knowledge repository is already used in transformation process. This information is in form of images, templates or documents that need to be load in PMIS.

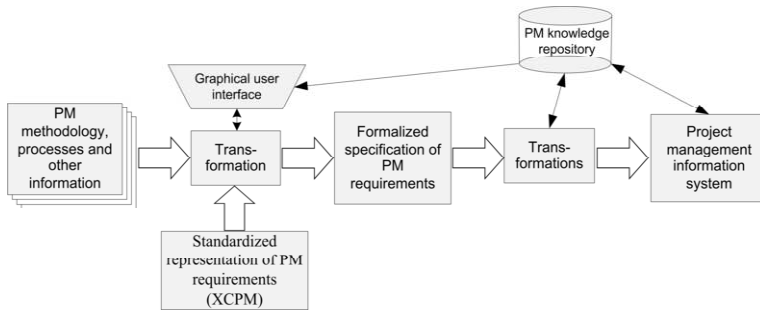


Figure 1. An approach for configuration of PMIS

1.2. Enterprise and Workflow System Configuration

To our knowledge, there are no papers addressing configuration of PMIS according to requirements of specific PM domain, but there are different approaches for configuration of workflow systems and enterprise applications such as ERP systems. Both systems focus on the automation of business process and information sharing, but workflow systems are process-centric and ERP systems are data-centric [11]. PMIS is similar to workflow system, but it is concentrated to a specific domain – PM.

Approaches of enterprise information systems and ERP implementation and configuration are based on standardized business process description with models. For example, model-driven implementation approach uses model driven architecture models transformation [12]. Object-process based approach uses ERP system model and enterprise requirement model alignment [13]. The model-driven configuration approach uses configurable event-driven process model [14]. The holistic approach uses three perspectives: management, business process analyst and technical analyst [15]. Each of these perspectives requires a model at different level of elaboration, for example, a business process analyst uses the configurable event-driven process model, but a technical analyst the workflow model [15].

A second opportunity of configuration description is XML which is widely used in workflow systems. For example, MS SharePoint workflows are stored in XML workflow markup file (XOML) [16]. XML process definition language (XPDL) is a widely used format for process definition exchange between workflow systems [17]. The configurable workflow model is developed for YAWL (Yet Another Workflow Language) and XML document is used for configuration description [18].

2. Project Management Concept Model

The XCPM schema is developed on the basis of the PMCM [8]. The PMCM is a composition of different concepts relevant to PM. It has been developed on the basis of existing conceptualizations of the PM domain, including XML schemas used in PM applications, PM ontologies and PM methodologies.

Each of these existing conceptualizations includes specific information, but no one is comprehensive enough that it can be used as basis for development of XCPM. XML schemas used in PM applications (PMXML [9] and MS Project XML [10]) define only operational PM data. PM ontologies (PROMONT [19], PMO [20] and others) focus on PM knowledge while data and process related aspects are represented only at very high

level of abstraction. Knowledge and process are described in PM methodologies (e.g. PMBOK [2], PRINCE2 [3], MSF [5], and RUP [4]) though this description is unstructured and, in the case of specific methodologies, represent just a single viewpoint of PM. For that reason, a separated concept model has been created for each of these sources.

The concept model integration has been performed to obtain the comprehensive concept model of the PM domain. XML schema integration principles and techniques have been used for integration of the concept models (PMCM integration algorithm is shown in Figure 2) [8]. The model integration consists of the following main tasks: pre-integration, conforming, merging and restructuring [21]. Development of individual concept models is performed in the pre-integration phase. During the conforming phase conflict detection and resolution are performed [21]. Three common types of the conflicts are naming, semantic, and structural [22]. Main types of conflict resolution transformations are entity/attribute, entity/relationship, and attribute/subtype equivalence [21]. Related concepts are identified and a single federated concept model is produced in the merging phase [21]. Four types of merging transformations are used, namely, entity merge, introduction of is-a relationships, addition of union, and addition of intersection [21]. During restructuring quality of the federated concept model is improved. Four types of restructuring transformations are used – redundant attribute removal, optional attribute removal, generalization of attributes, and redundant relationship removal [21]. Conforming and merging processes are executed in an iterative manner in each stage adding one concept model. After the technical integration process, restructuring is performed.

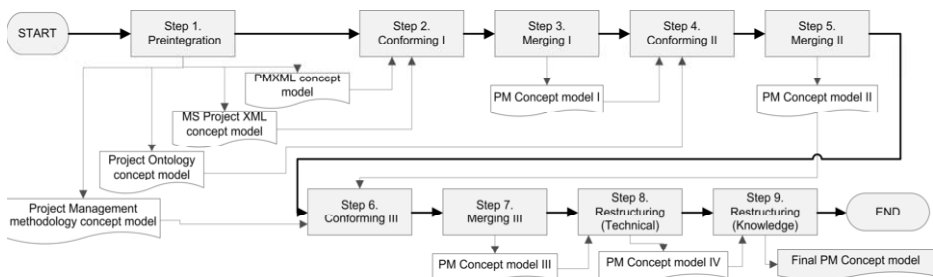


Figure 2. PM concept models integration algorithm

After completing the integration process, the PMCM is obtained. The model defines all main entities and relationships present in the PM domain. A fragment of the PMCM is shown in Figure 3. The central entity of the model is *Project*. This entity is directly or transitionary related to sets of entities describing project planning and project controlling (e.g. *RequestChanges*). The *KnowledgeArea* entity defines PM knowledge areas describing knowledge and processes needed for successful PM. It represents a generic case, while there are several entities representing specific knowledge related entities such as *IntegrationManagement* and *ChangeManagement*. *KnowledgeAreas* entity summarizes general *KnowledgeArea* and specific knowledge entities. *KnowledgeAreas* entity is part of *ProjectLifeCycle* entity that includes information about PM organization. The *Process* entity represents various PM processes.

The proposed conceptualization of the PM domain has been developed in parallel with development of reference model Ref-Mod^{PM} by Ahlemann [23]. The purpose of this model is to facilitate design of PM software, set-up of the surrounding organization

system and definition of PM software requirements. This model is based on a references project life-cycle model. PMCM allows to specify a custom project life cycle. Main entity of Ref-Mod^{PM} is data structure called *Initiative* and it is generalization of entities used in different existing PMIS and is any form of action that has defined start and end dates. Contrary, the PMCM contains entities from different sources and PM methodologies are used as primary source. Both models include descriptive elements such as project, WBS, roles, resources, calendars etc. However, Ref-Mod^{PM} only partially represents processes and collaborations among actors involved in PM. These both aspects are very important in configuration of PMIS. The Ref-Mod^{PM} includes elements from all knowledge areas of PMBOK though representations of these elements have not been elaborated.

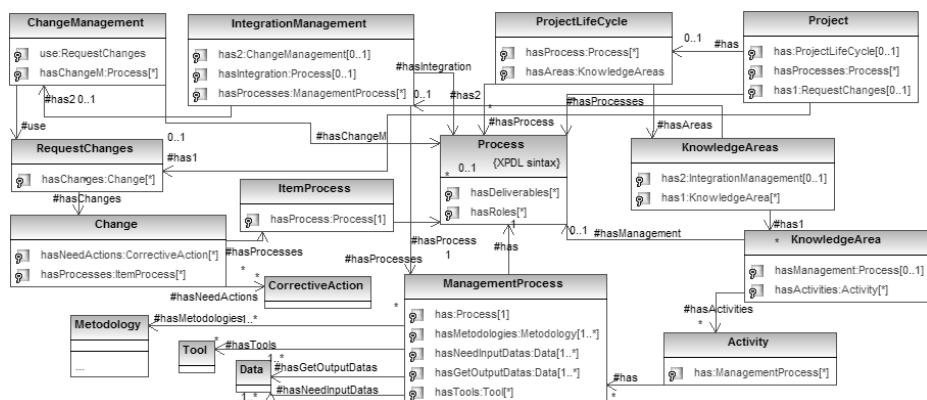


Figure 3. Fragment of PMCM

3. Structure of XCPM

The forward engineering approach is used to develop XCPM from PMCM that includes three development stages: development of conceptual schema, development of logical schema and development of physical schema [24]. In this case, PMCM is the conceptual scheme. The logical schema or XCPM is obtained using information in PMCM and transformation rules. In this case, the physical schema already includes certain data specifying particular project management methodology according to the structure provided by XCPM.

The following rules are used to design the XCPM schema from PMCM:

- XCPM schema main top-level elements are entities in PMCM, which are directly related to the Project entity;
- Relations between elements are organized either as links or sub-elements;
- Abstract elements are added to ensure project specific information description and attribute definition (e.g. OtherElement, Attribute);
- Similar type entities are merged in one element (e.g. KnowledgeArea);
- Both the configuration and data elements are defined.

The XCPM schema includes elements to describe all entities described in the PMCM. The schema elements are divided into two groups: descriptive elements and management elements. Main XCPM schema elements are shown in Figure 4.



Figure 4. Overview of XCPM main top-level elements

Main descriptive top-level elements are listed and described in Table 1. Sets of elements are subordinated to each of the top-level element. Each set of elements describes or collects any information about an area of project or PM.

Table 1. XCPM descriptive top-level elements

Element	Description
<i>Calendars</i>	All calendars in project such as project, resource , and contract calendars
<i>Roles</i>	Roles in project, it skills, works and related reports.
<i>Metrics</i>	Project metrics and measurements
<i>Deliverables</i>	Project deliverables and it acceptance measurements
<i>WBS</i>	Project work breakdown structure
<i>Limitations</i>	Different project assumptions and constraints
<i>Milestones</i>	Project milestones
<i>Activities</i>	Project activities or task and describe it related WBS components, predecessor or successor activities, performance reviews, baselines, assumptions, requirements
<i>Schedule</i>	Information about activity and milestone order
<i>Resources</i>	Project human and material project resources
<i>OBS</i>	Organization breakdown structure and it authorities
<i>EnvironmentFactors</i>	Different project environment factors such as project area, type, organization type, related standards and other project attribute
<i>RBS</i>	Risk breakdown structure
<i>RiskRegister</i>	Project risks, it measurements, processes, corrective actions, responsible roles and other information
<i>CBS</i>	Project contracts, tasks, sellers, deliverables, documentation and progress reports
<i>QualityRequirements</i>	Project quality requirements, it measurements, criteria, corrective actions, deliverables and standards
<i>ProgressReport</i>	Progress reports in project
<i>LessonLearnedLog</i>	Lessons learned information
<i>PerformanceReviews</i>	Earned values, variances and forecasts
<i>ProjectManagement Documents</i>	PM documents and plans
<i>IssueLog</i>	Project issues, corresponding corrective actions and processes
<i>ChangesRequests</i>	Project changes, corresponding corrective actions and processes
<i>Assignment</i>	Resource or role assignment to activity or milestone
<i>Communication Requirements</i>	Communication requirements, meetings, communication objects and other commutation information
<i>OtherElement</i>	Define different specific project information

ChangesRequests is one of the descriptive elements that define changes, corresponding corrective actions and processes. The change request definition in PMCM definition was given Figure 3 and its representation in the XCPM schema is shown in Figure 5. All specific attributes for *Change* is defined with *Attribute* (Figure 6). *Processes* and *CorrectiveActions* are sub-elements groups for *Change*.

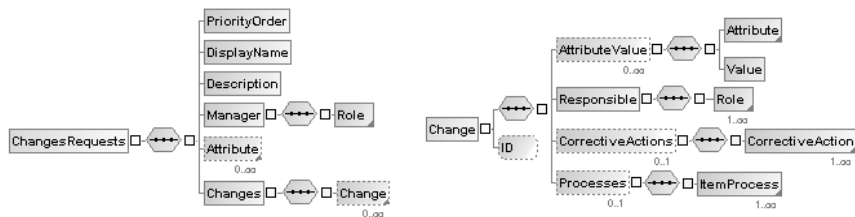


Figure 5. Fragment of XCPM: *ChangesRequests* and *Change*

The *OtherElement* element (Figure 6) is used to define different specific project information, for example, in a software maintenance project it is used to define configuration items, testing documentation and reviews. This element allows describe specific project information items, its attributes and store values.

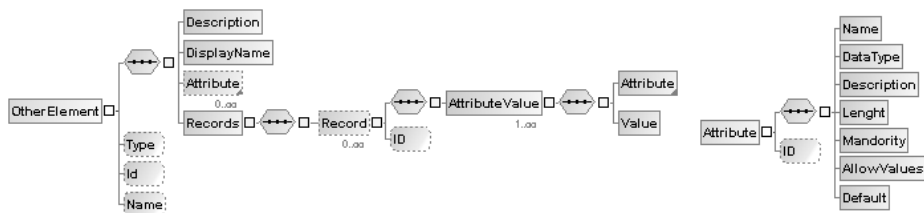


Figure 6. Fragment of XCPM: *OtherElement* and *Attribute*

ProjectLifeCycle and *Processes* main top-level elements are included in the management group. Elements in this group describe project management activities and processes and uses elements from descriptive group.

The *Processes* element describes dynamic PM processes such as change control, risk management, planning/monitoring processes and risk/change/issue injected processes. The process itself is defined as a workflow. This workflow in the XCPM schema is included using exiting XML description of workflows – XPDL2.1 [17]. XPDL includes description of activities, transactions, participants and artefacts. Any element is used in *Process* definition to enable including fragments of XPDL document. *ProjectDeliverables* and *UsedRoles* elements are used to ensure linking of XPDL process participants and artefacts with project roles and deliverables. *ProcessInputDatas*, *ProcessOutputDatas* and *ProcessTools* elements define relation between input/output data, tools and XPDL document parts. To describe process local deliverables and measurements *ProcessDeliverables* element is used. Definition of the *Process* entity in PMCM was shown in Figure 3 and its definition in the XCPM schema is shown in Figure 7.

The *ProjectLifeCycle* element is used to define main project process and *KnowledgeAreas* (Figure 8). Every knowledge area can be described using *KnowledgeArea* element. In PMCM classic knowledge areas are distributed as separate entities. There are three knowledge area entities in PM concept model fragment shown in Figure 3: *IntegrationManagement*, *ChangeManagement* and unspecified *KnowledgeArea* entity. All these entities are related with *Process* and

ManagementProcess entities. These entities differ in that each is used to work with different PM area entities. Consequently, it is sufficient that *KnowledgeArea* element will be provided with attributes which indicate involved descriptive elements.

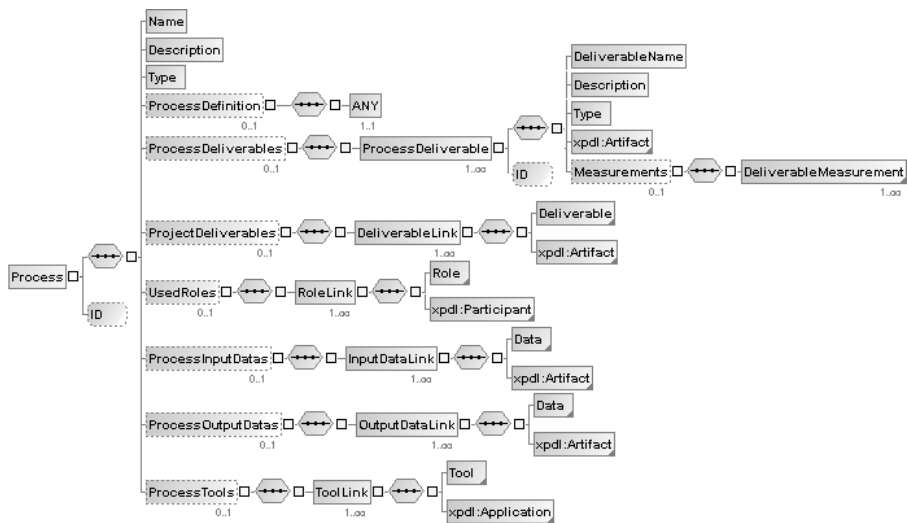


Figure 7. Fragment of XCPM: *Process*

The *KnowledgeArea* element (Figure 8) includes description of used methodologies, knowledge area process, PM activities (Figure 9), PM processes, input/output datas, tools and processes. All PM knowledge dynamic processes are described using the *Process* element (Figure 7)

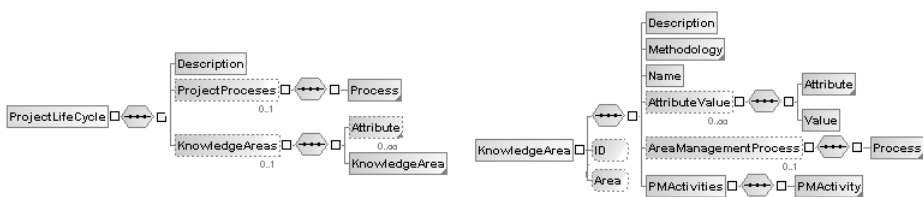


Figure 8. Fragment of XCPM: *ProjectLifeCycle* and *KnowledgeArea*

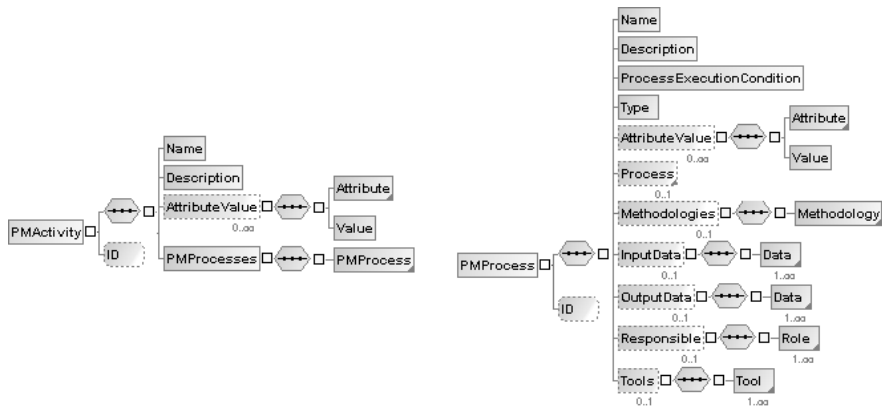


Figure 9. Fragment of XCPM: *PMActivity* and *PMProcess*

4. Prototype of PMIS Configuration

Change control (CC) is used as a representative example to demonstrate the use of the XCPM schema in configuration of PMIS.

CC is a part of project integration management and it is described in various methodologies/standards (PMBOK, PRINCE2, ISO 9000, COBIT, ITIL, RUP, MSF, SWEBOK). CC provides a standardized process to identifying, documenting, approving or rejecting, and controlling changes to the project baseline [2]. The CC process includes following six activities: change identification, analyze, evaluate, planning, implementation, review and close [25]. The change request or change is central element in CC. Every change may have the following status: submitted, evaluated, rejected, approved, change made, cancelled, verified and closed [26]. Changes are identified in the change identification activity, but its attributes are described during the change analysis activity. In our example change submit is an event that injects change analysis process, which requires determination of technical feasibility and costs/benefits. This process is shown in Figure 10. Other CC activities are only descriptive for this prototype.

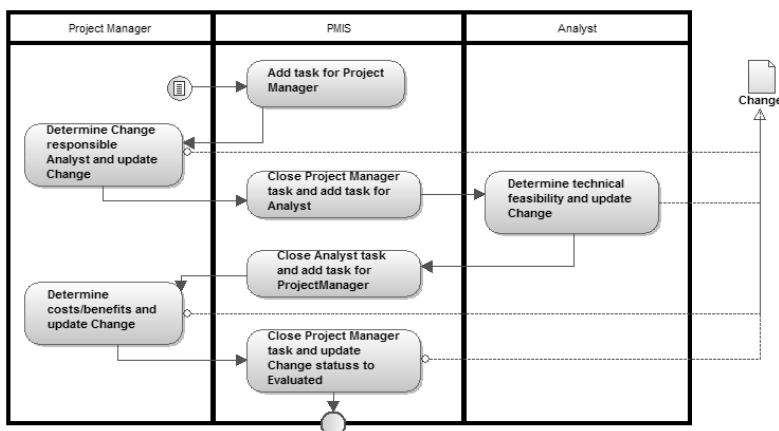


Figure 10. Change analysis process

The above description provides an informal representation of CC. In order to use this information in configuration of PMIS, it is structured according XCPM. An XML document specifying CC in the standardized manner is obtained as the result of this transformation (in this case, the document contains only elements relevant to CC and other elements are omitted because of limited scope of the example). This document is referred as CCXML. Main elements of CCXML are shown in Figure 11 and more detail description of change requests, change analysis process and CC area are in Appendix.

Change is a standard element in PM so it is defined using element *ChangeRequests*. Attributes for each change are defined using the *Attribute* elements. Change analysis process is described using *Processes* element group. CC area and activities are defined using *KnowledgeArea* and related elements.

The last configuration step is loading the CCXML document into PMIS. The loading is performed using software specific transformation scripts. In the example, the Microsoft Project Server is configured according the defined CC process. Change list and workflow setting of Microsoft Project Server is shown in Figure 12. Mapping

between CCXML document (Figure 11) and PMIS configuration (Figure 12) is shown in Table 2.

```
<Project name='Example_SB_2010_2'> ...
<ChangesRequests>...</ChangeRequests>
<Processes>
<Process id="AnalyzeChange">...</Process>
</Processes>
<ProjectLifeCycle>
<Description/>
<KnowledgeAreas>
<KnowledgeArea id="CC">...</KnowledgeArea>
</KnowledgeAreas>
</ProjectLifeCycle> ...
</Project>
```

Figure 11. Main elements of CCXML document

With *ChangesRequests* defined Changes is added as list in PMIS (Number 1 in Figure12), but it attributes is defined as columns for Changes (Number 2 – 5 in Figure 12). *AnalyzeChange* process after change submitting is identified by change analysis action described with *KnowledgeArea* element. Change processes in PMIS can be viewed in Changes workflow settings (Number 6 in Figure 12).

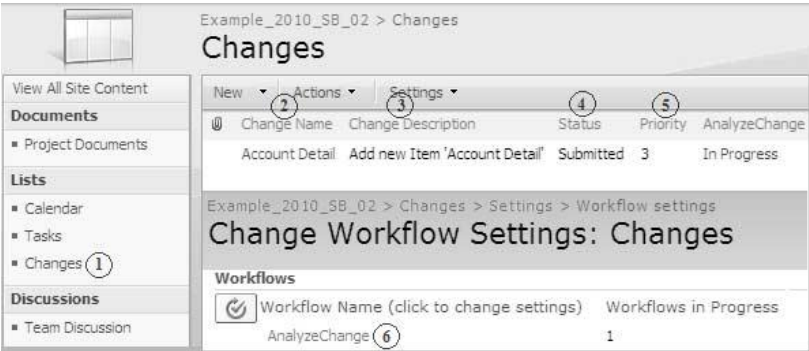


Figure 12. Example PMIS configuration according CCXML document

This CCXML document or it parts can be reused in other projects and also loaded in other PMIS.

Table 2. PMIS elements mapping with CCXML document

Number (Figure 12)	PMIS element	XPath to CCXML document (Figure 11 and Appendix)
1	List <i>Changes</i>	Project/ChangesRequests/DisplayName
2	Field <i>Change Name</i>	Project/ChangesRequests/Attribute [@id=ChangeName]
3	Field <i>Change Description</i>	Project/ChangesRequests/Attribute [@id=ChangeDescr]
4	Field <i>Status</i>	Project/ChangesRequests/Attribute [@id=ChangeStatus]
5	Field <i>Priority</i>	Project/ChangesRequests/Attribute [@id=ChangePriority]
6	Workflow for Change	Project/ ProjectLifeCycle/ KnowledgeAreas/ KnowledgeArea [@id= CC]/ PMAactivities/ PMActivity [@id= ChangeAnalyze]/ PMProcesses/ PMProcess [@id=ChangeAnalyze]
	<i>AnalyzeChange</i> process	Project/ Processes/ Process[@id=AnalyzeChange]

5. Conclusion and Future Work

This paper described the XML-based specification of the PM domain referenced as XCPM. XCPM includes PM information and requirements for PMIS and is built on the basis of the comprehensive PM concept model. It allows description of PM data, process and knowledge. PMIS configuration description and also data storage is possible in the schema.

The main application area of this XML-based specification is automated configuration of PMIS. XCPM ensures standardized description of PMIS configuration that further using software specific transformation scripts is loaded in PMIS. XCPM allows to load the same project definition XML document in different PMIS and reuse its parts in other projects.

Other application opportunity of XCPM is PMIS configuration knowledge store. Already prepared and used XCPM documents stored in the database is a part of the PM knowledge repository. This collected information is used as knowledge and guidelines in development of a new XCPM document.

Definition of the XCPM schema is a part of the proposed PMIS configuration approach. Definition of automatic transformation scripts, validation of XCPM schema and knowledge support in the configuration process are next steps of future research.

Acknowledgments

The author would like to acknowledge Dr.sc.ing. Jānis Grabis for his scientific guidance and support.

This work has been supported by the European Social Fund within the project «Support for the implementation of doctoral studies at Riga Technical University».

References

- [1] Raymond, L., Bergeron, F.: Project management information systems: An empirical study of their impact on project managers and project success. *International Journal of Project Management*, vol. 26, pp.213-220, Elsevier Ltd (2008)
- [2] Project Management Institute: Guide to the Project Management Body of Knowledge (PMBOK) Fourth edition. PMI Publication, Newtown Square (2008)
- [3] Hedemen, B., Heemst, G.V.van, Fredriksz, H: Project Management Based on PRINCE2 – PRINCE2 Edition 2005. Van Haren Publishing, Zaltbommel (2007)
- [4] Kruchten, P.: The Rational Unified Process: An introduction Third edition. Addison-Wesley, Boston (2004)
- [5] Turner, M.S.V.: Microsoft Solutions Framework Essentials: Building Successful Technology Solution. Microsoft Press, Washington (2006)
- [6] Bērziša, S.: The Baseline Configuration of Project Management Information System. *Scientific Journal of Riga Technical University, Information Technology and Management Science*, vol. 5, no. 40, pp. 59-65, RTU Publisher House (2009)
- [7] Bērziša, S., Grabis, J.: An Approach for Implementation of Project Management Information Systems. In: Papadopoulos, G. A., Wojtkowski, W., Wojtkowski, G. (eds.) *Information System Development: Toward a Service Provision Society*, pp. 423-431. Springer (2009)
- [8] Bērziša, S.: Towards an XML Scheme for Configuration of Project Management Information Systems: Conceptual Modelling. Grundspenkis, J., Kirikova, M., Monolopoulos, Y., Novickis, L. (eds.) *Advances in Databases and Information Systems Associated Workshops and Doctoral Consortium of the 13-th East-European Conference, ADBIS 2009, Riga, Latvia, September 7-10, 2009 Proceedings LNCS*, vol. 5968, pp. 229-237. Springer (2010)

- [9] Cover Pages, Project Management XML Schema (PMXML), <http://xml.coverpages.org/projectManageSchema.html>
- [10] XML Structure for Microsoft Office Project 2003, [http://msdn.microsoft.com/en-us/library/aa210619\(office.11\).aspx](http://msdn.microsoft.com/en-us/library/aa210619(office.11).aspx)
- [11] Cardoso, J., Bostrom, R.P., Sheth, A.: Workflow Management Systems and ERP Systems: Differences, Commonalities, and Applications. Information Technology and Management, vol. 5, pp. 319-338, Springer Netherlands (2004)
- [12] Gugerdl, P., Gaillard, G.: Model-Driven ERP Implementation. 8-th International Conference on Enterprise Information Systems (ICEIS2006), Workshop on Model-Driven Enterprise Information Systems (MDEIS), Paphos, Cyprus, May 23 – 27, INSTICC Press (2006)
- [13] Soffer, P., Golany, B., Dori, D., Aligning an ERP system with enterprise requirements: An object-process based approach, Computers in Industry, vol. 56, pp. 639–662, Elsevier B.V. (2005)
- [14] Recker, J., Mendling, J., Aalst, W.M.P van der, Rosemann, M.: Model-driven enterprise system configuration. Advanced Information System Engineering, LNCS, vol. 4001, pp. 369-383, Springer Berlin/Heidelberg (2006)
- [15] Dreiling, A., Rosemann, M., Aalst, W.M.P. van der, Sadiq, W.: From Conceptual process models to running systems: A holistic approach for configuration of enterprise system processes. Decision Support Systems, vol. 45, pp. 189-2007, Elsevier (2007)
- [16] Holliday, J., Alexander, J., Julian, J., Robillard, E., Schwartz, B., Ranlett, M., Attis, J.D., Buenz, A., Rizzo, T.: Professional SharePoint 2007 Development, Wiley Publishing, Indianapolis (2007)
- [17] Workflow Management Coalition, XPD L Support and Resources, <http://www.wfmc.org/xpdl.html>
- [18] Gottschalk, F., Aalst, W.M.P van der., Jansen-Vullers, M.H., La Rosa, M.: Configurable Workflow Models. International Journal of Cooperative Information Systems, vol. 17, pp. 177-221, World Science Publishing (2008)
- [19] Abels, S., Ahlemann, F., Hahn, A., Hausmann, K., Strickmann, J.: PROMONT - A Project Management Ontology as a Reference for Virtual Project Organizations. On the Move to Meaningful Internet Systems 2006, OTM 2006 Workshops, LNCS, vol. 4277, pp. 813-823. Springer, Berlin/Heidelberg (2006)
- [20] Ruíz-Bertol, F.J., Dolado, J.: A Domain Ontology for Project Management. In: Berki, E., Nummenma, J., Sunley, I., Ross, M., Staples G. (eds.) Software Quality Management XV: Software Quality in the Knowledge Society, pp. 317-326. British Computer Society (2007)
- [21] McBrien, P., Poulouvassilis, A.: A formalisation of semantic schema integration. Information Systems, vol.23, No.5, pp. 307-334. Elsevier Science Ltd, Great Britain (1998)
- [22] Aleksandraviciene, A., Butleris R.: A Comparative Review of Approaches for Database Schema Integration. Advances in Information Systems Development, pp. 111-122, Springer US (2007)
- [23] Ahlemann, F.: Towards a conceptual reference model for project management information systems. International Journal of Project Management, No.27, pp. 19-30., Elsevier (2009)
- [24] Fong, J., Cheung S.K.: Translating relational schema into XML schema definition with data semantic preservation and XSD graph. Information and Software Technology, vol. 47, pp. 437-462, Elsevier (2005)
- [25] Change management (engineering), [http://en.wikipedia.org/wiki/Change_management_\(engineering\)](http://en.wikipedia.org/wiki/Change_management_(engineering))
- [26] Change control process example, http://www.processimpact.com/process.../change_control_process.doc

Appendix: Fragments of CCXML document

Changes are described using ChangesRequests and Attribute elements:

```

<Roles>
  <Role id="rolePM"> <Name> Project Manager </Name> ... </Role>
</Roles>

<AttributeList>
  <Attribute id="ChangeName">
    <Name> Change Name </Name>
    <DataType> Text </DataType>
    <Description/>
    <Lenght> 100 </Lenght>
    <Mandority> Yes </Mandority>
    <AllowValues/> <Default/>
  </Attribute>
  <Attribute id="ChangeDescr"> ... </Attribute>

```

```

    <Attribute id="ChangeStatus"> ... </Attribute>
    <Attribute id="ChangePriority"> ... </Attribute>
</AttributeList>

<ChangeRequests>
  <DisplayName> Changes </DisplayName>
  <PriorityOrder> ASC </PriorityOrder>
  <Description/>
  <Manager> <Role id="rolePM"> </Manager>
  <Attribute id="ChangeName"/>
  <Attribute id="ChangeDescr"/>
  <Attribute id="ChangeStatus"/>
  <Attribute id="ChangePriority"/>
  <Changes/>
</ChangeRequests>

```

AnalyzeChange process is described using *Process* element:

```

<Process id="AnalyzeChange">
  <Name> AnalyzeChange </Name>
  <Description/>
  <Type> Management <Type>
  <ProcessDefinition>
    <Package ... >
      <PackageHeader/>
      <WorkflowProcesses>
        <WorkflowProcess Id="ChangeControl" Name="Change Control">
          <ProcessHeader/>
          <Participants>
            <Participant Id="PMIS" Name="PMIS"> <ParticipantType Type="SYSTEM"/> </Participant>
            <Participant Id="ProjectManager" Name="Project Manager"> <ParticipantType Type="ROLE"/>
            </Participant>
            <Participant Id="Analyst" Name="Analyst"> <ParticipantType Type="ROLE"/> </Participant>
          </Participants>
          <Artefacts> <Artefact ID="Change" Name="Change"> ... </Artefact> ... <Artefacts>
          <Activities>
            <Activity Id="Action1" Name="Add task for Project Manager">
              <Implementation> ... </Implementation> <Performer>PMIS</Performer>
            </Activity>
            <Activity Id="Action2" Name="Determine Change responsible Analyst and update Change">
              <Implementation> ... </Implementation> <Performer>ProjectManager</Performer>
            </Activity> ...
          </Activities>
          <Transitions>
            <Transition From="Action1" Id="ChangeControl_tra1" To="Action2"/>
            <Transition From="Action2" Id="ChangeControl_tra2" To="Action3"/> ...
          </Transitions>
        </WorkflowProcess>
      </WorkflowProcesses>
    </Package>
  </ProcessDefinition>
  <ProjectDeliverable/>
  <ProjectDeliverable/>
  <UsedRoles>
    <RoleLink> <Role id="rolePM"/> <Participant Id="ProjectManager"/></RoleLink> ...
  </UsedRoles>
  <ProcessInputDatas>
    <InputDataLink> <Data ID="ChangeItem"/> <Artefact ID="Change"/> </InputDataLink>
  </ProcessInputDatas>
  <ProcessOutputDatas>
    <OutputDataLink> <Data ID="UPDChangeItem"/> <Artefact ID="Change"/> </OutputDataLink>
  </ProcessOutputDatas>
  <ProcessTools/>
</Process>

```

Change control knowledge area is described using *KnowledgeArea*, *PMActivity* and *PMProcess* elements:

```

<PMProcessList>
  <PMProcess ID="ChangeAnalyze">
    <Name> Change Analysis </Name> <Description/>
    <ProcessExecutionCondition> Change status = Submitted </ProcessExecutionCondition>
    <Type/>
    <Process ID="AnalyzeChange">
    <Methodologies/>
    <InputData> <Data ID="ChangeItem"/> </InputData>
    <OutputData> <Data ID="UPDChangeItem"/> </OutputData>
    <Responsible/>
    <Tools/>
  </PMProcess>
</PMProcessList>

<PMActivityList>
  <PMActivity ID="ChangeIdentification"> ... </PMActivity>
  <PMActivity ID="ChangeAnalyze">
    <Name> Change Analysis </Name>
    <Description> Determine technical feasibility and costs/benefits </Description>
    <PMProcesses>
      <PMProcess ID="ChangeAnalyze"/>
    </PMProcesses>
  </PMActivity>
  <PMActivity ID="ChangeEvaluate"> ... </PMActivity>
  <PMActivity ID="ChangePlanning"> ... </PMActivity>
  <PMActivity ID="ChangeImplementation"> ... </PMActivity>
  <PMActivity ID="ChangeReviewClose"> ... </PMActivity>
</PMActivityList>

<ProjectLifeCycle>
  <Description/>
  <KnowledgeAreas>
    <KnowledgeArea ID="CC" Area="Change control">
      <Name> Change control </Name>
      <Description/>
      <PMActivities>
        <PMActivity ID="ChangeIdentification"/>
        <PMActivity ID="ChangeAnalyze"/>
        <PMActivity ID="ChangeEvaluate"/>
        <PMActivity ID="ChangePlanning"/>
        <PMActivity ID="ChangeImplementation"/>
        <PMActivity ID="ChangeReviewClose"/>
      </PMActivities>
    </KnowledgeArea>
  </KnowledgeAreas>
</ProjectLifeCycle>

```

Combining Functional and Nonfunctional Attributes for Cost Driven Web Service Selection

Mārtiņš BONDERS, Jānis GRABIS, Jānis KAMPARS

*Institute of Information Technology, Riga Technical University,
1 Kalku Street, Riga, LV – 1658, Latvia, martins@iti.rtu.lv*

Abstract. Selection of appropriate web services is an important step in development of composite applications. Quality of Service (QoS) data characterizing nonfunctional properties of candidate web services are usually used in web service selection. However, functional characteristics, which are difficult to measure, are equally important. Quantitative evaluation of functional characteristics is possible in the case of development of composite decision-making applications. This paper elaborates a web service selection method, which accounts for both functional and nonfunctional characteristics of candidate web services. The functional characteristics are evaluated according to decision-making results yielded by the composite application. A linear programming model is used as the web service selection method and the cost of developing and operating the composite application is used as a selection metric. Application of the proposed web service selection method is demonstrated by developing a composite application for a vehicle routing problem, where web services are required to provide spatial data necessary for the decision-making algorithm. Experimental results show that accounting for functional characteristics substantially affects web service selection results.

Keywords. Web services, QoS, vehicle routing

Introduction

Composite applications use external services to gain access to vast data and processing resources. Web services relying on such open technologies as XML, SOAP, WSDL, REST and others are often used as building blocks for composite applications. That allows developing standards-based, scalable and efficient applications. However, properties of composite applications directly depend upon characteristics of external services used and environmental factors, which in the case of public networks, exhibit high degree of variability. Therefore, selection of appropriate and reliable services is of major importance. Multiple methods have been elaborate for selection of such services from the set of candidate services providing similar functionality [1,2].

These methods often use Quality-of-Service (QoS) measurements to evaluate services. These measurements including reliability, security, trust and execution cost are mainly concerned with nonfunctional characteristics of services. However, service consumers are equally concerned about both functional and nonfunctional characteristics of services and there have been attempts to expand the QoS concept in the case of web service selection by defining it as: “The degree to which a system,

component or process meets customer or user needs or expectations” [1]. This definition includes evaluation of both functional and nonfunctional requirements. Unfortunately, formal evaluation of functional characteristics in the framework of web service selection is more difficult than evaluation of nonfunctional characteristics. A functional quality of service approach [2] uses similarity measures to identify interoperable web services. A QoS-aware service selection algorithm includes functional requirements in the model though these are represented only by a binary variable indicating either complete satisfaction or complete dissatisfaction of the requirement [3]. Generally, evaluation of functional characteristics either involves expert judgment or has limited resolution. Additionally, the service selection is inherently multi-objective problem. Preemptive optimization and weighting based approaches are usually used to account for different often contradicting objectives. However, these methods again rely on judgmental appraisal of relative importance of each selection criterion.

In order to address these issues, functional characteristics of web services also should be evaluated quantitatively and considered jointly with nonfunctional QoS characteristics using a universal metric to represent different selection criteria. The objective of this paper is to elaborate a quantitative approach for selecting web services according to both functional and nonfunctional requirements and to demonstrate application of this approach in development of composite decision-making applications. The universal metric used in the service selection is the total cost of building and operating the composite application. This approach is similar to the total cost of ownership approach [4] but the distinctive feature is evaluation according to both functional and nonfunctional attributes. Additionally, nonfunctional attributes such as response time and reliability are expressed in terms of their impact on cost of operating the composite application.

Application of the proposed approach is demonstrated using a decision-making application, which is used to determine optimal routing of multiple postal vehicles. Two main modules of this application are the routing algorithm and the data retrieval module. The data retrieval module uses external services to obtain data about distances between customers and travel time. Several web services are evaluated to determine the service giving the best results according to both functional and nonfunctional characteristics.

The remaining part of the paper is organized as follows. Section 2 describes web services characteristics most frequently used in web service selection. The approach for selecting web services according to both functional and nonfunctional requirements is elaborated in Section 3. The sample application and experimental results are discussed in sections 4 and 5, respectively. Section 6 concludes.

1. Web Service Selection Criteria and Methods

Web service selection has attracted large interest from academicians and practitioners alike. A number of web service selection methods have been elaborated and several typical QoS measurements used in web service selection can be identified by analyzing these methods. Table 1 surveys selected web service selection methods. All these methods are multi-criteria selection methods because web service selection is an essentially multi-criteria problem. Analytical Hierarchical Process (AHP) is the most frequently method used. Usually this method is used together with other methods.

Different methods from the artificial intelligence domain such as fuzzy algorithms and artificial neural networks are also frequently considered to account for factors, which are difficult to express analytically.

Table 1. Overview of Web service selection and approach.

Publication	Approach
[5]	QoS-aware composite service binding approach based on Genetic Algorithms
[6]	The Artificial Neural Network
[7]	Decentralized trust and reputation mechanisms for peer-to-peer based Web service systems
[8]	GFS (Goodness-Fit Selection algorithm) based on QoS prediction mechanism in dynamic environments
[9]	Fuzzy-based UDDI with QoS support
[10]	QoS Consensus Moderation Approach (QCMA) in order to perform QoS consensus
[11]	A query-by-example representing Web service descriptions and queries as vectors
[12]	Analytic hierarchy process (AHP) approach, Stateless and stateful session beans are applied in the CSPMP matrices
[13]	QoS meta-model as the basis for the QoS and AHP modeling
[14]	An QoS-aware services selection model based on LINear programming techniques for Multidimensional Analysis of Preference
[15]	Select services by considering two different contexts: single QoS-based service discovery and QoS-based optimization of service composition. Based on QoS measurement metrics, this study proposes multiple criteria decision making and integer programming approaches to select the optimal service
[16]	Convergent population diversity handling genetic algorithm (CoDiGA)
[17]	Analytic Hierarchy Process (AHP) and the Brown–Gibson (BG) methods
[18]	Graph transformation rules to describe the semantics of Web services. These rules provide precise semantic specifications needed for an automated service discovery in a visual and intuitive way

The literature reviews suggests that there are two main categories of attributes used in web services selection: QoS properties and business properties category [19,20]. The QoS properties category may be divided into two sub categories: execution and security properties.

Table 2 lists nonfunctional QoS characteristics considered in selected papers. Response time, accessibility and availability are the most universally used characteristics in the QoS properties category. Cost is the most frequently used business related characteristic.

Table 2. Overview of QoS characteristics used in web service selection.

Source	QoS characteristics												
	Execution			Security				Strategic					
	Response/execution time	Accessibility	Compliance	Successability	Availability	Encryption	Authentication	Access control	Cost	Reputation	Organization arrangement	Payment method	Monitoring
[5]	x	x			x				x				
[21]	x	x		x	x				x		x		
[9]	x	x		x	x								
[10]	x								x				
[19]	x	x	x	x	x	x	x	x	x	x	x	x	x
[18]	x		x			x	x	x	x				
[22]	x	x			x		x	x					

The cost of using web service includes multiple positions. The total cost of ownership approach implies that cost of using information technology solutions include such costs as software and hardware, installation and integration of hardware and software, warranties and licenses, technology research, migration cost [23]. In the case of web services, many of these positions could be avoided, for instance, software and hardware costs. However, the licensing and integration costs are still relevant.

2. Functional Evaluation of Web Services

Nonfunctional QoS characteristics usually used in service selection have been identified in the previous section. However, users are equally concerned with functional characteristics, which are unique to a particular application, and representation of functional characteristics is more difficult in formal service selection models. Two approaches can be used to evaluate web service conformance to functional requirements: 1) expert evaluation; and 2) quantitative performance measures. Existing service selection methods mainly use expert evaluations. Finding quantitative performance measures characterizing functional requirements is difficult. One approach could be using business key performance indicators though these indicators give only indirect measurements, and they are affected by different side-effects. Decision-making applications form one class of applications for which quantitative performance measurements can be determined because functionally these applications are required to produce specific performance indicators. For instance, if composite application is used in vehicle routing then the main functional requirement is finding a route between two locations. In this case, alternative services can be evaluated by comparing travel times returned by the services.

The proposed web service selection approach evaluates candidate services according to both functional and nonfunctional requirements for decision-making applications, where functional requirements can be evaluated by direct quantitative performance measures. The overall service selection approach is shown in Figure 1.

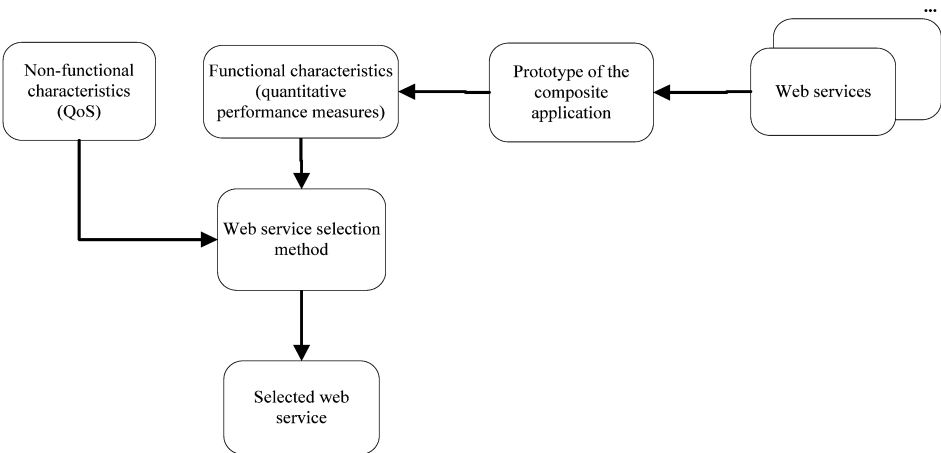


Figure 1. Web service selection according to functional and nonfunctional characteristics

3. Web Service Selection Model

It is assumed that requirements for a composite application relevant to web service selection are given by N functional requirements referenced by index i and M nonfunctional requirements referenced by index j . There are L alternative web services $\mathbf{S} = (s_1, \dots, s_L)$ satisfying the requirements to some degree. QoS data $\mathbf{G}_l = (g_{l1}, \dots, g_{lM})$ characterize how well the l th service performs according to the j th nonfunctional requirement and quantitative performance measures $\mathbf{H}_l = (h_{l1}, \dots, h_{lN})$ characterize how well the l th service performs according to the i th functional requirement. \mathbf{G}_l can be determined according to data provided by the service provider or using simulation studies. In order to evaluate \mathbf{H}_l , a prototype composite application is built. A variant of the prototype P_l is developed for each candidate web service. Values of \mathbf{H}_l are determined by executing the corresponding prototype P_l . A web service selection method takes \mathbf{G}_l and \mathbf{H}_l as input data and finds the best service using an appropriate selection algorithm.

As indicated in Section 2, different selection methods can be used to choose the most appropriate services given their quantitative characteristics. In this paper, a linear programming model is used as the selection method. In order to combine different functional and nonfunctional criteria, a universal optimization metric is used. This metric characterizes the cost of building and operating the composite application depending upon web services used in its development. It is computed as

$$TC = \sum_{l=1}^L \left(\sum_{i=1}^N c_i^f h_{li} + \sum_{j=1}^M c_j^{nf} g_{lj} \right) X_l, \quad (1)$$

where X_l indicates whether the l th service is selected and c_i^f and c_j^{nf} are parameters characterizing costs associated with functional and nonfunctional requirements, respectively. The web service selection objective is to minimize the total cost by selecting the most appropriate service:

$$TC(\mathbf{X}) \rightarrow \min \quad (2)$$

The optimization is performed subject to the following constraints

$$\sum_{l=1}^L X_l = 1, \quad (3)$$

$$h_i \leq u_i^f, \forall i, \quad (4)$$

$$g_j \leq u_j^{nf}, \forall j, \quad (5)$$

$$X_l \in \{0, 1\}, \forall l. \quad (6)$$

The first constraint implies that only one service is selected. Eqs. (4) and (5) filters out services having worse characteristics than specified threshold u_i^f and u_j^{nf} for functional and nonfunctional requirements, respectively. The key issue in solving this linear programming model is finding values of the cost parameters c_i^f and c_j^{nf} .

This model is equivalent to a simple weighted-linear scoring though the linear programming formulation is more flexible and can be augmented with other parameters and constraints.

4. Application Example

Application of the proposed web service selection approach is demonstrated using an example of development of composite application for routing postal service vehicles.

4.1. Problem Description

In the case of routing postal service vehicles, the postal service receives a daily list with planned parcel deliveries and needs to schedule its fleet of vehicles to fulfill these requests. Each planned delivery is defined by a customer address and a delivery time window. The delivery time window specifies the earliest and latest possible delivery time. All vehicles depart from the single depot and also need to return to the depot within the normal operating hours. There are about a thousand delivery requests per day, and there are about 20 delivery vehicles. The postal service aims to minimize delivery costs and to minimize delivery time what directly influences a number of vehicles needed to serve all customers. Figure 2 shows the graphical representation of the vehicles routing problem. It can be observed that a separate route is created for each vehicle, and each customer is assigned to exactly one route.

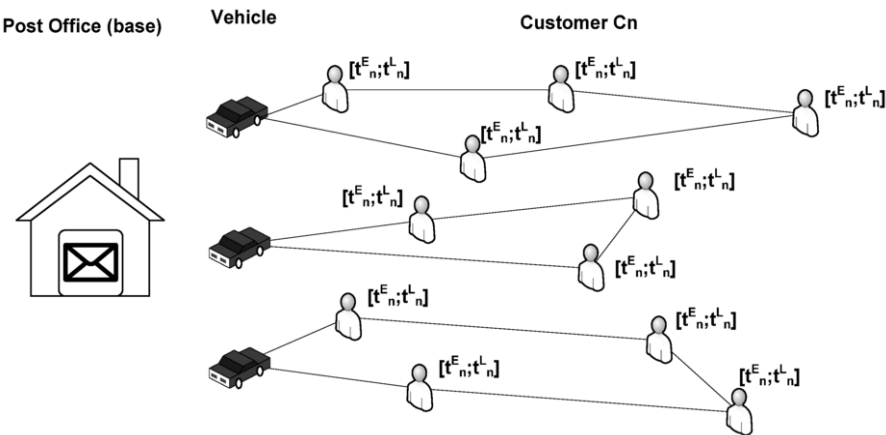


Figure 2. Postal vehicle routing example, where $[t_n^E; t_n^L]$ represents the delivery window

4.2. Composite Decision-Making Application

A composite decision-making application (Figure 3) is developed in order to solve the vehicles routing problem described above. This application receives the list with planned deliveries and number of vehicles in the postal service fleet as input data. The list of planned deliveries shows addresses of customers and the delivery time window. The application determines the number of vehicles necessary to serve all customers and a delivery route for each vehicle. This delivery route shows the sequence of customer visits and the scheduled arrival time at each customer. The application also yields the total travel time and travel distance for all vehicles. Thus, the main functional requirements for the composite vehicle routing application are determining number of

necessary vehicles, creating the delivery route for each vehicle and finding the total travel time and distance.

Two main components of the composite application are the vehicle routing module and data gathering module. The vehicle routing module is responsible for finding delivery routes using the input data provided. The data gathering module is responsible for retrieving data necessary for finding the delivery routes. These data include input data and additional data retrieved using web services. The necessary additional data are distances between customers and travel times from one customer to another. They can be retrieved from cartographical web services. There are a large number of such cartographical web services. In order to obtain the most efficient delivery routes, the most appropriate web service should be selected. This web services is selected using the service selection approach described in Section 3. The quantitative performance measures used for functional requirements are: 1) number of vehicles; 2) route length in kilometers; and 3) driving time in hours. The QoS measures used for nonfunctional requirements are: 1) response time; and 2) percentage of dropped requests (corresponding to the successability characteristic defined in Section 2). Additionally, the ease of integration factor is considered. This factor characterizes development efforts needed to integrate a candidate web service into the composite application and influences cost of integration (licensing cost is not considered because free web services are used).

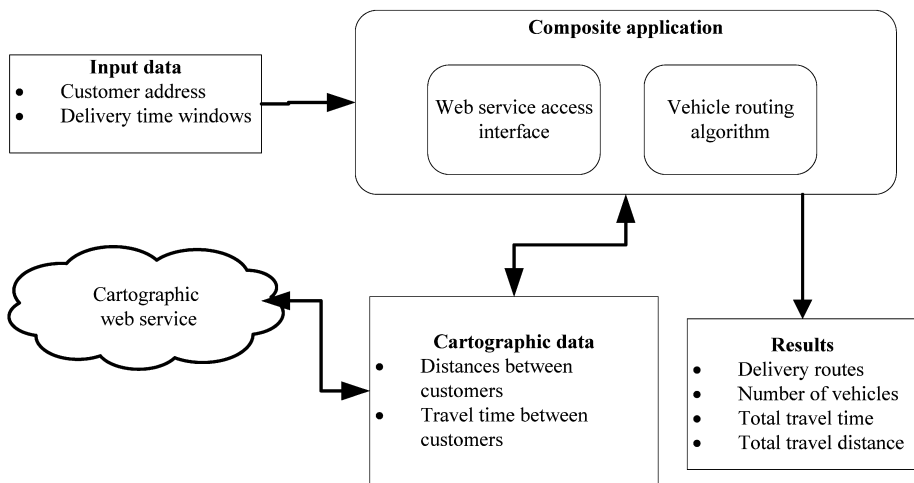


Figure 3. The composite vehicle routing application.

4.3. Routing Algorithm

The routing algorithm is one of key modules of the composite application. The heuristics vehicles routing algorithm with time windows developed by Solomon [24] is this application. The main difference between this algorithm and routing algorithms used in cartographical web services is that this algorithm creates routes for multiple vehicles simultaneously and accounts for restrictions imposed by time windows.

The heuristic algorithm is used because it is computationally more efficient than optimal algorithms. It initializes a route for the first vehicle by assigning a customer to the route. A customer nearest to the first customer in the route is identified using the

distance measurements retrieved from web services. If time window constraints can be satisfied then this customer is added to the route. If time window constraints are violated then other nearest customers are checked. If none of customers can be added to the current route then a new route is initialized and the next vehicle is scheduled for deliveries. This process is continued until all customers have been included in one of the routes.

4.3.1. Estimation of Cost Parameters

The cost parameters associated with functional and nonfunctional criteria identified in Section 4.2 are estimated. The cost parameter c_1^f for the number of vehicles criterion is expressed as in Eq. (7) (in this case it is specific to a particular service though simple transformations could eliminate this dependency)

$$c_{11}^f = a_1 \left(1 - \frac{h_1^b}{h_{11}} \right) \tau_1, \quad (7)$$

where a_1 is hourly rate for using a vehicles, τ_1 is the number of times vehicle routing is performed during the planning horizon and h_1^b is the benchmark according which to evaluate performance of the given web service. The benchmark is a predefined best possible performance according to the given performance criterion. It is used because from the functional perspective there is a base cost level, which does not depend upon a kind of application or service used.

The cost parameters c_2^f and c_3^f are computed similarly using appropriate usage rate parameters and benchmarks, where a_2 is a charge per kilometer traveled a_3 is a charge per hour spend on the road.

The response time criterion is expressed in terms of costs by calculating time necessary to gather all input data and taking into account the hourly rate for data gathering:

$$c_1^{nf} = a_4 \tau_4 d, \quad (8)$$

where a_4 is hourly data gathering rate, τ_4 is a number of times data gathering is repeated during the planning horizon and d is the number of data points requested. Assuming that failed request should be executed once again, the percentage of requests dropped has similar effect to data gathering cost.

Additionally, integration of each service into the composite application incurs certain development costs. For the ease of integration criterion, the cost parameter c_3^{nf} is defined as monthly development rate and the value of criterion g_3^{nf} is calculated by transforming the number of source code lines into the effort measured using personmonth [25].

5. Experimental Results

Experimental studies are conducted to demonstrate service selection for the composite vehicle routing application. A set of 100 addresses representing customer locations was generated. The postal office depot address is chosen manually, and customer addresses are randomly generated within 150 km around the depot. Three candidate variants of

the composite application are implemented. Each variant uses a different cartographic web service while other parts of the application remain the same. The cartographic web service should be able to calculate travel time and distance between any two customers defined by their address.

Table 3 lists values of parameters used in the service selection model. These parameters are used to calculate cost according to functional and nonfunctional factors. It is assumed that the composite application is executed to obtain delivery routes every working day during a year while data gathering is performed weekly to account for possible changes in cartographic data.

Table 3. Value of parameters

Parameter	Value
Data gathering repetitions (τ_4)	52
Routing execution repetitions (τ_1)	253
Development rate (c_3^{nf} , EUR/month)	8385
Data gathering rate (a_4 , EUR/h)	28
Data points (d)	2500
Vehicle cost (a_1 , EUR/day)	35
Travel distance cost (a_2 , EUR/km)	0.2
Travel time cost (a_3 , EUR/h)	6

Three functionally similar publicly available web services are analyzed. They are preselected according to the strategic QoS characteristics and are referred as S_1 , S_2 and S_3 , respectively. To account for uncertainty in evaluation of QoS measurements and for variation in routing results due to input data used, ten lists with delivery requests are generated by randomly drawing 50 addresses from the initial set of addresses. A time window for each delivery request is also generated.

Each variant of the composite routing applications is executed for each list with delivery requests. QoS data (i.e., response time and percentage of dropped requests) are accumulated during the execution and quantitative performance measures are determined. The integration cost is determined by counting the number of source code lines (SLOC) in the web service integration code and transforming SLOC into integration effort.

Values of the nonfunctional criteria and the corresponding nonfunctional and integration costs are given in Table 4. The nonfunctional cost is calculated as a function of the average response time and percentage of dropped requests and it represents cost of data gathering. It can be observed that the percentage of dropped requests has minor impact on nonfunctional costs. S_3 has the lowest nonfunctional and integration cost among services considered. Values of the functional criteria and the corresponding functional costs are given in Table 5. The table also gives the total cost of using the particular web service. Assuming that all services satisfy minimum criteria threshold level, the candidate service yielding the smallest TC is also the best service. The lowest functional cost is obtained using S_1 , which also gives the smallest travel distance and travel time. Despite having the highest nonfunctional and integration cost, this service also yields the lowest total cost suggesting that the functional criteria dominate the nonfunctional criteria for this service selection problem. Figure 4 confirms this observation.

Table 4.Nonfunctional cost and integration cost for candidate web service.

Service	Average response time (g_1, s)	Dropped requests ($g_2, \%$)	Nonfunctional cost	Integration code SLOC	Integration Cost
S_1	4.59	0	4641	158	4024
S_2	4.53	0.41	4599	176	4024
S_3	4.23	0.11	4281	86	2012

Table 5.Functional cost for candidate web services.

Service	Average number of vehicles needed	Average travel distance (km)	Average travel time (h)	Functional cost	Total Cost (TC, EUR)
S_1	11.00	2704	59.52	2656	11322
S_2	11.50	2903	64.82	25198	33822
S_3	10.70	2808	60.59	6886	13180

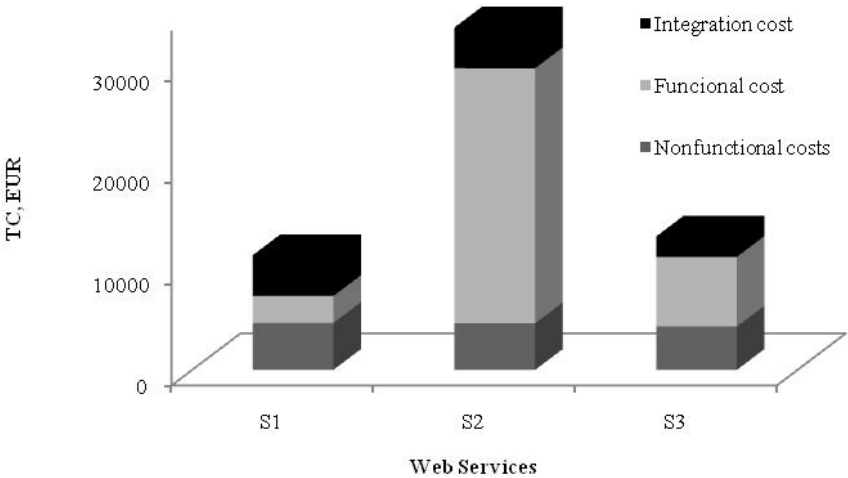


Figure 4. The cost breakdown for candidate web services

However, the relationship between importance of the nonfunctional and functional criteria depends upon usage characteristics of the composite application. If data gathering is performed only once a year then the nonfunctional cost is negligible (Figure 5). If data gathering is repeated every time the composite application is used for routing then the share of the nonfunctional cost may exceed 70%. For the given problem, data do not change as frequently to warrant data gathering every day though this issues is of major importance for real-time and near real-time applications. The share of nonfunctional cost also substantially depends upon number of data points.

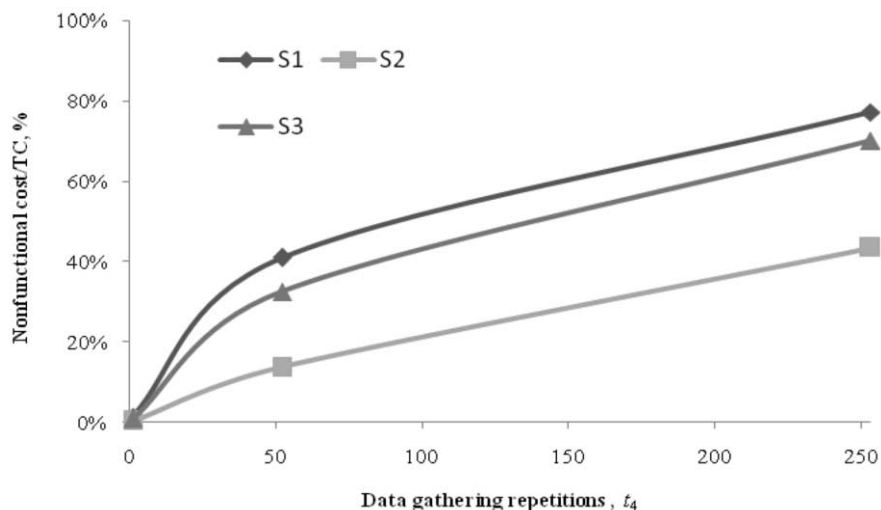


Figure 5. Share of nonfunctional costs depending upon data gathering repetitions

6. Conclusion

An approach for selecting web services has been elaborated. The main distinctive features of this approach is accounting for both functional and nonfunctional characteristics of candidate web services. That has been achieved by using quantitative performance measures to quantify the functional characteristics. However, evaluation of these measures is challenging, and, in this paper, it has been restricted to class of composite decision-making applications. The obtained experimental results for the vehicle routing problem show that web service selection substantially depends upon accounting for functional characteristics. During implementation of the sample composite application, it has been observed that strategic QoS characteristics have major impact on pre-selection of web services.

One shortcoming of the proposed approach is necessity to implement a variant of the composite application for each web service to be evaluated. However, replacing a web service component for the properly design application is not a very effort consuming task though there could be situations when developing multiple variants is prohibitive. A training data set is also necessary to perform functional evaluation of candidate web services. In the case of vehicle routing application, a field validation of the obtained results is also necessary because it is possible that the presumably best web service gives inaccurate estimates of travel distances and travel times.

The cost of developing and operating the composite applications is used as a service selection metric, which allows combining multiple selection criteria. A shortcoming of this approach is that some of the costs can only be determined indirectly. An alternative is using a multi-objective mathematical programming model as the web service selection model though evaluation of relative importance of each criterion is subject to judgmental appraisal.

Acknowledgment

This work has been supported by the European Social Fund within the project „Support for the implementation of doctoral studies at Riga Technical University”.

References

- [1] Huang Angus F.M., Ci-Wei Lan, Stephen J.H. Yang, An optimal QoS-based Web service selection scheme, *Information Sciences: an International Journal* archive Volume 179 , Issue 19, Pages: 3309–3322, 2009.
- [2] Buhwan J., Hyunbo C., Choonghyun L., On the functional quality of service (FQoS) to discover and compose interoperable web services *Expert Systems with Applications* 36, 5411–5418, 2009.A.N.
- [3] Dimitrios T., Ioanna R., Efstathios S., QoS-aware service evaluation and selection, *European Journal of Operational Research* Volume 191, Issue 3, Pages 1101–1112, 2008.
- [4] Garfamy, R.M., A data envelopment analysis approach based on total cost of ownership for supplier selection, *Journal of Enterprise Information Management*, v 19, n 6, 662–78, 2006.
- [5] Massimiliano Gerardo C., Di P., Raffaele E., Maria L. V., A framework for QoS-aware binding and re-binding of composite web services, *Journal of Systems and Software* Volume 81, Issue 10, October 2008, Pages 1754–1769 Selected papers from the 30th Annual International Computer Software and Applications Conference (COMPSAC), Chicago, September 7–21, 2006.
- [6] Haibin C., Xiaohui H., Qiyang C., A novel intelligent service selection algorithm and application for ubiquitous web services environment, *Expert Systems with Applications* Volume 36, Issue 2, Part 1, Pages 2200–2212, 2009.
- [7] Yao W., Vassileva J., A Review on Trust and Reputation for Web Service Selection, *Department of Computer Science University of Saskatchewan*, Vol. 0, 25, 2007.
- [8] Li M, Huai J., HuiPengGuo, An Adaptive Web Services Selection Method Based on the QoS Prediction, *School of Computer Science and Engineering, Beihang University, Beijing, China, IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology – Workshops*, 2009.
- [9] Wang Hei-Chia, Chang-Shing Lee, Tsung-Hsien Ho, Combining subjective and objective QoS factors for personalized web service selection, *Expert Systems with Applications* 32 571–584, 2007.
- [10] Wei-Li Lin, Chi-Chun Lo, Kuo-Ming Chao, Muhammad Younas, Consumer-centric QoS-aware selection of web services, *Journal of Computer and System Sciences* 74, 211–231, 2008.
- [11] Crasso M., Alejandro Z., Marcelo C., Easy web service discovery: A query-by-example approach, *Science of Computer Programming* 71, 144–164, 2008.
- [12] Jiachen H., Daizhong Su, Integration of Web Services technology with business models within the total product design process for supplier selection, *Computers in Industry* 57, 797–808, 2006.
- [13] Chen Wu, E., Chang, Intelligent Web Services Selection based on AHP and Wiki, *IEEE/WIC/ACM International Conference on Web Intelligence*, 2007.
- [14] Angus F.M. Huang, Ci-Wei Lan, Stephen J.H. Yang, An optimal QoS-based Web service selection scheme, A.F.M. Huang et al. / *Information Sciences* 179, 3309–3322, 2009.
- [15] Menasc'eDaniel A., R., Honglei, G., Hassan, QoS management in service-oriented architectures, *Performance Evaluation* 64, 646–663, 2007.
- [16] Yue Ma, Zang C., Quick convergence of genetic algorithm for QoS-driven web service selection, *Computer Networks* 52, 1093–1104, 2008.
- [17] Sun Y., He S., Jack Y. Leu, Syndicating Web Services: A QoS and user-driven approach, *Decision Support Systems* 43, 243–255, 2007.
- [18] Hausmann J. H., Heckel R., Lohmann M., Towards Automatic Selection of Web Services Using Graph Transformation Rules, *Faculty of Computer Science, Electrical Engineering and Mathematics University of Paderborn Warburger Str. 100 33098 Paderborn*, 2003.
- [19] Badr Y., Abraham A., Biennier F., Grosan C., Enhancing Web Service Selection by User Preferences of Non-Functional Features, *Proceedings of the 2008 4th International Conference on Next Generation Web Services Practices* table of contents, Pages: 60–65, 2008.
- [20] Cardoso J., Miller J., A. Sheth, J. Arnold, Modeling quality-of-service for workflows and Web service processes, *Web Semantics: Science, Services and Agents on the World Wide Web* Volume 1, Issue 3, Pages 281–308, 2004
- [21] XuanV., T., Tsuji H., Masuda R., A new QoS ontology and its QoS-based ranking algorithm for Web services, *Simulation Modelling Practice and Theory* Volume 17, Issue 8, Pages 1378–1398, *Dependable Service-Oriented Computing Systems*, 2009.

- [22] Diamadopoulou V., Makris C., Panagis Y., Sakkopoulos E., Techniques to support Web Service selection and consumption with QoS characteristics, *Journal of Network and Computer Applications* 31, 108–130, 2008.
- [23] Aziz, M.H., Ong Con Nie, Jesse Chan Mei Yam, Lee Chang Wei, TCO Reduction, *Communications, 2003. APCC 2003. The 9th Asia-Pacific Conference*, 1147 - 1151 Vol.3, 2003.
- [24] Solomon M. Marius, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Northeastern University, Boston, Massachusetts, December 1985.
- [25] Barry Boehm, Bradford Clark, Ellis Horowitz, Ray Madachy, Richard Shelby, Chris Westland, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Engineering*, (1995) 57-94

WSRF Usage for BPM and CEP Systems Integration

Normunds BLUMBERGS, Maksims KRAVCEVS¹,

University of Latvia, Riga, Latvia

{NormundsBlumbergs@gmail.com, Maksims.Kravcevs@lu.lv}

Abstract. The paper describes principles, architecture and design of a general solution for business process management systems integration with complex event processing (CEP) engines. As CEP query standards are still in early development we define a general abstract CEP Query Model and corresponding distributed infrastructure: CEP Network to abstract from specific CEP engines. We propose integration between BPEL Engines and CEP Network using intermediary bidirectional CEP Service. Key part of the solution is to represent CEP Network queries as WSRF resources. Events from CEP Network are received using WS-Notification. This approach allows using standard BPEL and does not require any changes or extensions in BPEL Engine and designing tool. We compare the proposed approach with some existing alternatives. We present a prototype which implements key components of the proposed solution and provides integration between an open source CEP Engine Esper and BPEL Apache ODE engine.

Keywords. CEP, BPEL, BPMS, WSRF, WS-Notification.

Introduction

Complex Event Processing (CEP) has lately become a Business Process Management (BPM) aspect next to Business Intelligence (BI) and Business Activity Monitoring (BAM). CEP fills the gap between BAM and BI: compared to BAM where only limited data on process are analysed CEP provides possibility to analyse external data of various nature and complexity, compared to BI CEP adds possibility to react in near to real time. As per Gartner Research² Inc. CEP is now a must have for BPM tooling. Also Forrester Research³ Inc. identifies CEP as essential component of a BPM solution.

CEP has been a separate technology until identified as an essential component in BPM technology stack. The objective of CEP is to discover complex inferred events by analyzing and correlating other events in real time [1]. CEP engines can analyse large amounts of operational level events – events that don't have direct business value. After analysis CEP allows to infer events that have business value as compared to source events, therefore they can be used in business processes. There are no industry CEP query language standards currently. In the last couple of years there have been some initiatives to define it, but support from CEP engine vendors is not sufficient since query language features are one of differentiating factors between different CEP

¹ Research is supported by grant ESS 2009/82 from University of Latvia and European Social Fund contract no 2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044

² www.gartner.com

³ www.forrester.com

products. CEP market also is still stabilizing – new vendors are entering the market (e.g. Microsoft StreamInsight⁴, JBoss Drools Fusion⁵), multiple mergers (e.g. Sybase⁶ acquiring Aleri, Coral8).

Currently there are two main integration solutions between CEP and BPM systems – the first approach is to provide CEP engine as an internal component of a BPM solution, the other approach is to use different adapters for CEP Engine. In cases that company uses BPM solution that doesn't have CEP component, company could only use adapters (migration to a new solution is usually not an option), which don't support proper CEP query lifecycle. In this paper we consider alternatives for integration between different BPEL engines and CEP Engines.

Main integration design goals are to ensure loose coupling and be based on open standards. We propose such a solution and implement a prototype as a proof of concept.

BPEL language standard is one of the commonly used in BPM engines. To avoid being specific to one CEP engine, we need to define an abstraction for CEP engines and their event processing models. We offer a notion of CEP Network, which represents an infrastructure that can connect several CEP engines, their clients and sources of events. Such an approach doesn't introduce any limitations as in the simplest case CEP Network corresponds to a single CEP engine. We also define a general CEP Query model as an event processing model in CEP Network. As basis for CEP Network and query model we take common possibilities of some following used CEP Engines: Sybase Complex Event Processing⁷, Esper⁸, Tibco⁹ and StreamBase¹⁰.

Our integration solution between CEP Network and BPEL Engine is based on an integration pattern with an intermediary component – CEP Service that will provide native integration method towards both BPEL process and CEP Network. This approach doesn't require changes or extensions in the BPEL and CEP Network models. The key design approach is to represent CEP Queries in CEP Service as WSRF resources. Event schema would match WS Resource properties that allow consumer to access event information by querying interested properties.

WSRF specification provides notification mechanism on resource changes via WS-Notification. WS-Notification allows BPEL process to subscribe and receive events from CEP Network using standard mechanisms.

We implement a proof of concept prototype, which contains all key components of the proposed solution. The proof of concept is built using Open Source solutions – Esper CEP Engine is used as a specific engine for CEP Network, Apache MUSE¹¹ used for CEP service, Apache ODE¹² is used as BPEL engine. The solution is published as Open Source Project at Google Code¹³.

The paper is structured as follows. In the first section we provide some general information on web service standards and CEP concepts we use. The second section

⁴ <http://blogs.msdn.com/b/streaminsight/>

⁵ <http://www.jboss.org/drools/drools-fusion.html>

⁶ <http://www.sybase.com/>

⁷ <http://www.coral8.com/>

⁸ <http://esper.codehaus.org/>

⁹ <http://www.tibco.com/software/complex-event-processing/default.jsp>

¹⁰ <http://www.streambase.com/>

¹¹ <http://ws.apache.org/muse/>

¹² <http://ode.apache.org/>

¹³ <http://code.google.com/p/ebpm/>

describes introduced CEP Network and CEP Query model abstraction. The third section gives a general overview of proposed integration solution. The fourth section provides more details on the solution components and their functionality. The fifth section describes some alternatives and existing solutions that can be used for similar purpose. The final section describes the prototype implemented and used technologies in it.

1. Background

1.1. Web Services, WSDL and WS-BPEL Standards

SOA propagates approach to IT system composition from loosely coupled and distributed components with clear contract definition – services. One of common technologies to implement services that are based on SOA principles is to use web service standards. Web service standards define a standard mechanism for exposure and consumption of data and application logic over Internet protocols such as HTTP using xml and are de facto standards for SOA implementations.

WSDL [2] is an xml language for describing of web service. It contains abstract and concrete definitions of web service. Abstract definition describes web service interface names, operations that can be performed on them, messages used to perform these operations and data types used in messages. Concrete definition describes the location of the web service implementation and the way it can be accessed. WSDL for webservice contains complete information enough for client application to technically interact with the web service.

WS-BPEL [3] is an xml web service orchestration language that defines business processes interacting with web services. WS-BPEL is most widely supported process definition language by modern BPM systems. WS-BPEL contains elements for definition of complex process flow and it utilises invocation of web services as process activities. It is possible to extend WS-BPEL to support additional web service standards. Sure support for the standard should be built in into BPEL Engine where this BPEL process is deployed and also implementations of web services that are called also should support these standards. An example of one of such extensions could be the WS-Atomic Transaction support. Then in BPEL we can specify that invocation of several web services within the process should be in one distributed transaction. Most BPEL Engines will support this, and thus will ensure passing of necessary technical information and additional technical messages to web services called from BPEL process.

1.2. WS-notification

WS-Notification [4] is a web service standard that provides message notification pattern. It allows web services to subscribe to receive notifications from other web services. The standard consists of several specifications. We use WS-Topics that allow defining subscription topics and WS-BaseNotification that defines direct interaction between Notification Consumers and Notification Producers. A Notification Consumer is an endpoint, designated to receive notifications produced by a Notification Producer. In order to subscribe to these notifications Notification Consumer should send to a

Notification Producer a standard subscription request and implement a standard operation which Notification Producer will call back to send notifications.

1.3. WSRF, WS-Resource, WS-Resource Lifetime

WSRF (*Web Services Resource Framework*) [5] is a set of specifications that extends web service definition (which is stateless by default) with state. WS Resource is defined as a logical entity extending web service, uniquely addressable, having attributes and a lifecycle. WS Resource refers to Resource Properties Document that contains Resource Descriptor, which describes resource attributes. Clients can manipulate resources: query resource properties. WS-Resource Lifetime specification defines WS Resource lifecycle.

WSRF is typically used when it is necessary to model and provide some structured information dynamically, see for example [6] where BPEL process model and state is presented as WS Resource with properties according to BPEL metamodel.

Important for us is the additional possibility defined to support WS-Notification for resource acting as a Notification Producer. There are separate WS topics defined for particular attribute and for resource as a whole. Thus according to WS-Notification standard client may subscribe to the necessary topics defined for the resource. Once resource properties are changed the notification for relevant topics is sent to subscribed clients.

WS Resource is created when first client requires it. Clients need to send keep alive messages through WS-Resource Lifetime set expiration time mechanism. If WS Resource doesn't receive messages from any of the clients within certain time period, then it is destroyed.

1.4. CEP Languages & CEP Engines

CEP language is a language used by CEP engine to manipulate events. As an example of CEP languages could be Esper query language [7]. Currently there are no common CEP language standards, though some researches and attempts are ongoing in this direction, e.g. [8]. There are common possibilities that languages provide, for an overview see [9].

Some of the basic notions are: Event Type – data structure, content of the event, in most cases flat; Event Streams – grouping of events of the same type representing event flow; Event Patterns – certain relations between events of interest.

CEP language allows defining event patterns to be detected and manipulate events in event streams, also creating new events. Quite often CEP languages use SQL (*Structured Query Language*) like syntax with additional elements for event analysis and manipulation. As an example query that analyses the incoming share price stream and extracts events when share price is critically low (less than 40):

```
Insert Into StreamLowSharePrice //target stream
  Select Id, SharePrice // data from incoming stream
  From StreamSharePrice // incoming stream
  Where Price < 40.0 // condition on incoming stream events
```

The event types for the streams in this case

```
StreamLowSharePrice ( Id (String), SharePrice (Float) )
StreamSharePrice ( Id (String), SharePrice (Float) )
```

2. CEP Network

Currently CEP Engines use different event processing models and different languages. There are several researches that offer certain general common model, e.g. [10], but these models are not yet merged into one common model.

For the purpose to consider a general mechanism of CEP Engines and BPEL Engines integration we define CEP Network as a general abstraction of CEP infrastructure. We also define CEP Query model as a model for query execution in this infrastructure. Our proposal is based on analysis of existing CEP Engines and can be considered as a modification of metamodel proposed in [10] focusing on distributed environment.

2.1. Event Type Model

We do not consider Event Type representation. We find it fairly straightforward and technical task to convert Event Types between different technical data representations. In general we assume that it can be set as certain XML structure with generally arbitrary complex hierarchical structure.

2.2. Generalised CEP Query Model

Figure 1. CEP Query model provides a representation of CEP query which can be executed in distributed environment with multiple CEP Engines possible.

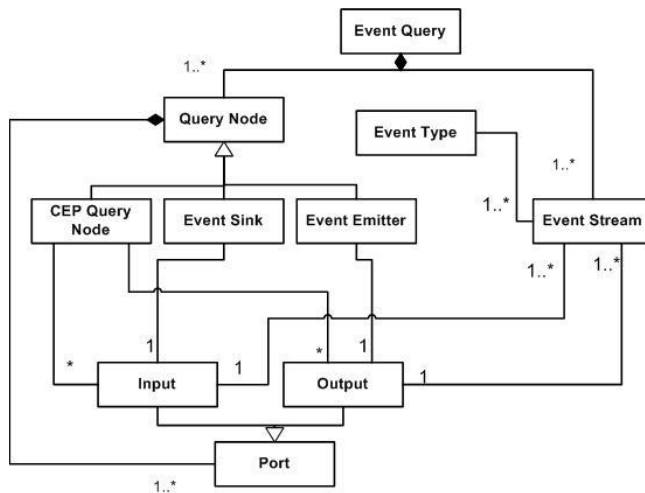


Figure 1. CEP Query model

We define Event Query as a set of Query Nodes connected by Event Streams. Event Streams are of defined Event Type and connect Query Nodes using Ports. Ports are either Inputs or Outputs. Each Event Stream connects to exactly one Input and one Output Port. There are 3 distinct Query Node types: Event Emitter nodes for events CEP Network receives from outside, Event Sink nodes for events CEP Network sends outside and CEP Query Nodes representing CEP Engines manipulating events. An Emitter has exactly one output port, a Sink has exactly one input port, and CEP Query Nodes may have multiple input and output ports.

2.3. Event Processing Network Infrastructure

Now we can define CEP Network as a run time infrastructure for CEP Query execution. The following diagram shows main elements of a CEP Network (Figure 2).

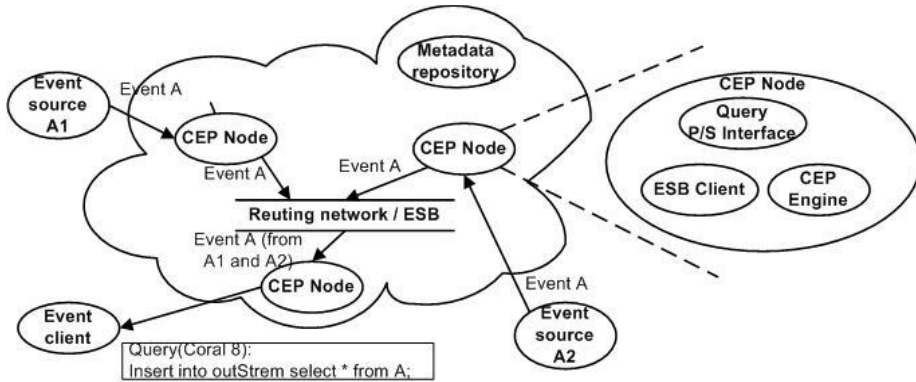


Figure 2. CEP Network runtime infrastructure

There are following components:

Event Sources – publish definite event types to the Network connecting to Event Emitters. The usual pattern is that each Emitter contains a Listener to which Event source can fire events.

Event Clients – subscribe to CEP Network by submitting CEP Query. CEP Network should ensure CEP Query deployment, processing and results delivery as CEP Events of the corresponding Event Type to Clients. The usual pattern is that Clients subscribe to CEP Network to receive events.

CEP Network – Dynamic infrastructure that ensures CEP Query execution in a distributed environment includes multiple CEP Nodes and controls event flow there.

CEP Network consists of 3 main elements:

- **CEP processing Nodes** – that ensure execution of CEP Queries and provide interface for communication with CEP Network (using publish /subscribe interface). CEP Engine is logically part of CEP Processing Node and connects to Event Routing network.
- **Metadata repository** – keeps information on the CEP Network components and configuration, event types registered and other.
- **Event Routing Network / Event Service Bus** – middleware that ensures communication between Processing Nodes and also Event Sources and Consumers. The ERN can be considered as a specific case of Enterprise Service Bus optimised for processing of high volumes of such events, and possibly realising certain operations on them.

In the simplest case CEP Network would consist of one CEP Processing Node – CEP Engine.

2.4. CEP Network Requirements

To complete CEP Network definition we need to define integration possibilities and quality of service we expect from it:

- Publish/subscribe (P/S) communication – that is usual pattern supported when interacting with CEP engines, which integrate to some messaging engine.
- In order event delivery – delivery of events within CEP Network and to the clients should not change the order of events.
- At least once delivery – events cannot be lost in CEP Network.

3. BPM and CEP Network Integration

In this chapter we describe the principles of the solution and give a high level overview of the proposed solution components.

The objective is to provide general and efficient integration mechanism between BPEL Engine and CEP Network. We should consider both - integration at design (or modelling) phase, when BPEL process and CEP queries are defined, and runtime, when they are executed.

The functional requirements for runtime integration are to ensure BPEL process to receive CEP events from CEP Network as well as post events to CEP Network. The functional requirements for design time are to provide an easy way to define the event exchange in BPEL and CEP Network design environments.

To ensure loose coupling we decide to keep BPEL Engine and CEP Network as independent as possible, we also require the integration solution to be based on open standards. We propose to implement an intermediary CEP Service that will provide native integration methods towards both BPEL process and CEP Network. The key decision is to represent CEP Queries as WSRF resources, which makes it easy to work with them in BPEL Process. The advantage of the approach is that it does not require changes or extensions in the BPEL and CEP Network models.

The design and execution of united BPEL/CEP process and solution components is shown on Figure 3. CEP Query is defined and deployed on CEP Network using CEP service client tool independently from BPEL process. Alternatively if historical event handling is not needed, then query could also not be deployed till WSRF resource is created; only new type would get registered. New event type becomes available for usage in BPEL process. Then once BPEL process is deployed, WS Resource gets created. Event delivery to consumer in this case BPEL process is achieved using WS Notification. CEP Service ensures WSRF standard and WS Notification support for CEP Query during its lifetime.

No changes in BPEL Engine or BPEL design tool are required. BPEL process is defined using standard BPEL design tool. Processing of events from CEP queries within BPEL process requires following code in BPEL Process: provide an operation where actual processing of event to be done, and put a subscription call on relevant WS Resource in CEP service, usually to be done at the start of the process passing there the operation. Additionally if WS Resource needs to be managed (deleted e.g. if it is not used by any BPEL process), then BPEL process must call Set Termination Time WS as per WS-Resource Lifetime to trigger resource lease extension. That can be done using standard BPEL possibilities, since WSRF is an extension over webservice.

The opposite direction in integration – BPEL process as Event Source to pass events to CEP Network is not shown on the diagram. Publishing to CEP Network from BPEL process is without durable subscription, but in fire and forget manner. CEP Service provides WS to publish event to any event type topic available in CEP

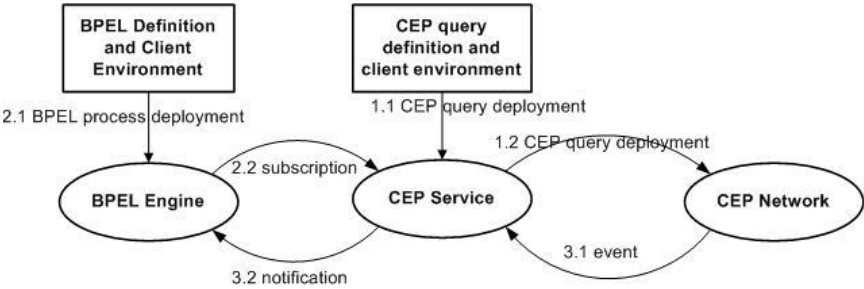


Figure 3. High level solution description

Network. BPEL process can publish events only to event types, not customized queries. Message correlation aspects between producers, consumers (both of which could be BPEL processes) can be managed at a higher functionality levels using event payload specific definition, which is a separate topic. Bi-directional communication between CEP Network and BPEL processes also allows inter process communications (which should be used if events need to be correlated again). For example – a business process exception occurs that is captured by CEP query, then corrective business process is executed, if corrective business process execution fails, it could send a new event back to CEP Network, a different query could trigger new event creation if corrective business process fails too often (this query processes process execution fail events), thus enabling higher level exception handling.

4. Detailed Solution Design

In this chapter we explain the design of the solution in more details. The following diagram Figure 4. Solution Components shows solution components and data flows.

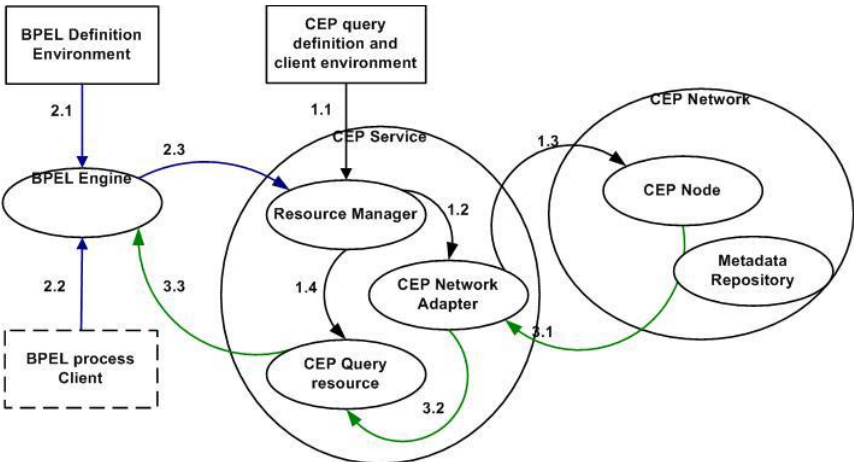


Figure 4. Solution Components

There are following components, design tools:

BPEL Definition Environment – tool for BPEL Process design. No changes are required.

CEP query definition and client environment – an environment to define and manipulate CEP queries. It uses CEP Service to pass CEP queries to CEP Network. CEP Query definition client connects to CEP service.

And runtime infrastructure:

BPEL Engine – BPEL process runtime environment. No changes are required.

CEP Network – components has been considered in details in chapter 3 CEP Network.

CEP service – integration application which consists of:

- Resource Manager – allows CEP service client to deploy CEP queries, based on CEP query definition creates CEP query resource components and connects them to CEP Network events (via CEP Network Adapter component). Also provide further access to manage created CEP Queries (update or destroy them).
- CEP Network Adapter – technical middleware used to subscribe to CEP Network events and forward them to the relevant CEP Query Resources.
- CEP Query resources – are created dynamically by Resource manager, implement WSRF standard interface. Each CEP Query corresponds to one WSRF resource. External clients (BPEL) can access them using WSRF standard operations and subscribe to receive WS notification on resource changes.

We describe now 3 main flows within this integrated infrastructure for the case when BPEL Engine needs to receive events from CEP Network.

Design and Deployment of CEP Query into CEP Network

1.1. User defines CEP query using some CEP query definition tool (not considered here) and inputs it to CEP service client. CEP service client connects to the Resource Manager in CEP Service and passes it there;

1.2. Resource manager validates the CEP query and starts its registration process. It requests CEP Network Adapter to deploy query to the CEP Network. In case query needs a lifecycle (should be used only when needed), then in this step CEP resource (and even the query itself) could not be created till requested by the client. In simple scenario based on the output event schema of the query a new CEP resource wsdl is generated with corresponding WS-Resource Metadata descriptor.

Resource properties are created according to the Event type. Resource structure is hierarchical, containing message queue. Message queue will allow keeping historical event information as well as provide latest event information. Message queue element will match actual event information. Each deployed query is uniquely identified by the output event type name that must be different from existing event types in the CEP Network (since queries that don't contain any processing logic – having input and immediately output node, match direct CEP Network events). Correspondingly WS Resource EPR (*end point references*) will be generated based on the event type name.

1.3. CEP Network Adapter subscribes to CEP Processing Node with the received CEP Query passing along the address where events to be sent from CEP Network;

1.4. Resource manager completes creation of CEP Resource.

Design and Deployment of BPEL Process

2.1. BPEL process is designed as usual using BPEL Designer. There should be following 2 steps defined in BPEL process:

- operation defined where processing of received event to be done;
- invocation of subscription call to CEP query resource.

We should note that they can be done in different processes. BPEL Process is deployed to BPEL engine, which validates and initialises it. Additionally if subscribed query has a lifecycle, then keep alive WS calls in a form of Set Termination Time must be performed.

2.2. BPEL instance is initiated. That is independent from deployment of process. BPEL Engine ensures different mechanisms for BPEL process invocation, e.g. providing web service interface to process, once BPEL process web service is invoked, the BPEL Process instance is created and executed. No extensions from our solution.

2.3. Executing BPEL process instance according to the BPEL code the subscription call to CEP query resource web service is executed.

Event Processing

3.1. CEP event fires in the CEP Network. CEP processing node passes it to all subscribers and thus sends it to CEP Network Adapter in the CEP service.

3.2. CEP Network Adapter delivers the event to CEP Query Resource;

3.3. CEP Query Resource using WSN sends the notification to the BPEL process calling corresponding operations.

Additionally since WS Resource is hierarchical and contains historical queue of events it's also possible for the BPEL process to retrieve all events using WS Resource provided WS query operations.

5. Alternative Solutions

We have considered several alternatives.

CEP Query Definition in BPEL Process

Using this approach we would be able to define CEP queries directly in the BPEL process. To enable such functionality it would require creating a custom BPEL extension. This would require adding support in BPEL for concrete CEP Engine languages. In case of a single vendor for the BPEL engine and CEP engine, it could be a preferred solution, since would provide users the ability to use a single environment to define full (including CEP queries) BPEL processes. On the other hand, in this case additional functionality would be needed to enable sharing of CEP queries (e.g. if there are multiple business processes sharing the same CEP query logic). In case of different vendors, this creates tight coupling from BPEL to specific CEP engine, which is not desired.

ESB Message Channel

Most obvious loosely coupled integration would be to rely on some ESB standard functionality for message routing. Adapter for CEP Network would put events as messages to ESB, and ESB would take care about their routing, as well as configuration of subscriptions would be done in ESB. Thus BPEL process would not

need to subscribe to event source. However in this approach we lose the definition of event type – which should be kept separately as well as in this case the control who and how consumes events becomes too distributed.

CEP Adapter Based

Most CEP engines provide abilities to input /output events through adapters. Adapters are of different kinds, e.g. file, databases, messaging systems. CEP engine doesn't handle adapter management or subscription handling. Meaning – dynamic adapters can't be created unless a special custom component managing adapters is created; they must be decided before query deployment. Other aspect is subscription handling – adapters don't support subscriptions, thus this support has to be defined separately anyway.

Observed Existing Solutions

We have also examined several existing solutions, which integrate CEP with BPEL or ESB. For reference purposes we also provide info on some complete solutions where CEP is fully integrated in this case no other integration needed.

Following open source solutions have been observed:

- JBoss Drools Fusion – CEP capabilities are integrated into a rules engine module Drools Expert, also CEP functionality is available as a separate module for standalone modelling scenarios, but to be used in BPEL processes it needs to be treated as sub entity to rules, which is limiting.
- Servicemix-Esper¹⁴. Solution integrates Apache Servicemix¹⁵ and Esper; event streams are realised as web services, Esper is encapsulated into component, which calls these webservice. This solution can't be used directly in BPEL, also Servicemix is a JBI container, therefore adds JBI environment dependency;
- Open ESB IEP [11], provides slightly different approach building CEP Support as part of ESB utilising ESB infrastructure for event exchange as ESB messages. This is attempt is to build in CEP Engine functionality into ESB. Main drawbacks include adding specific support to particular vendor solution, routing all CEP events through general purpose ESB, no direct support for BPEL process execution.

Commercial ones:

- ActiveVos¹⁶ – full solution. In ActiveVos it's possible to define CEP queries on-event actions that can start process execution. CEP is deployment time attribute of orchestrations, so processes are not altered. Event consumption in CEP is only supported for events emitted by BPEL execution engine.
- Sybase CEP – Sybase indirectly bought Coral8, thus internal CEP engine is the one from Coral8. Currently Sybase has integrated CEP engine into Financial Services Solutions, providing ability to use events easier through improved adapters, but still only through adapters.

¹⁴ <http://code.google.com/p/servicemix-esper/>

¹⁵ <http://servicemix.apache.org/home.html>

¹⁶ <http://www.activevos.com/>

- Integrated messaging based solutions: IBM Websphere Business Events¹⁷, Tibco Business Events. Main approach is a standard transformation between different messaging mechanisms. IBM Websphere Business Events for example offers adapters integration, transformations between adapter and BPEL process type definitions.

6. Solution Prototype

The main purpose of prototype is to demonstrate the complete integration scenario of CEP Network events usage in BPEL process. The prototype contains realisation of all components of integrated infrastructure based on Open Source products. Prototype source is published as open source project eBPM at Google Code¹⁸. Prototype components and their technologies are described below.

CEP Network Application

- Event Routing Bus – uses JMS, built on Apache ActiveMQ¹⁹
- Repository - HSQLDB²⁰ in memory database
- CEP Processing Nodes – custom developed Java RMI²¹ server which provides publish /subscribe (P/S) interface to CEP queries. CEP Engine encapsulated into processing nodes is Esper.
- Event Sources – post events to CEP processing Nodes using JMS

CEP Service Application

- CEP Service Application is built on top of Apache Muse²² WSRF standard implementation.
- Separate WSRF resource is created dynamically by Resource manager for each CEP Query. CEP Service uses RMI to connect to the CEP Processing Nodes publish subscribe interface. CEP service acts as a standard intermediary propagating client calls to WSRF resource to subscribe to CEP queries and passes back the events received by RMI.
- CEP Service Application is deployed to Apache Axis2²³ webservice container setup on Jetty²⁴ web server. Due to historical reasons Apache Muse service application is run on a separate Tomcat²⁵ application server. We have left this distributed CEP service and Muse service setup to demonstrate that components can be fully distributed.

¹⁷ <http://www-01.ibm.com/software/integration/wbe/>

¹⁸ <http://code.google.com/p/ebpm/>

¹⁹ <http://activemq.apache.org/>

²⁰ <http://hsqldb.org/>

²¹ <http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>

²² <http://ws.apache.org/muse/>

²³ <http://ws.apache.org/axis2/>

²⁴ <http://www.mortbay.org/jetty/>

²⁵ <http://tomcat.apache.org/>

CEP Service Management Client Application

This part of prototype has not been completed and thus not included to prototype. Instead CEP Query configuration for CEP Service is defined in CEP service configuration file and CEP Service implementation relies on Esper Query deployed directly to Esper Engine.

BPEL Process Environment

Apache ODE²⁶ 1.3.4 BPEL Engine deployed on on Apache Tomcat is used for BPEL process execution. Eclipse 3.6. Helios²⁷ has been used for BPEL Process definition. No customization of the BPEL tools has been needed.

Sample

Sample run on this setup consists of 2 Event sources passing events to the CEP Network with one Processing Node where Esper CEP Engine is setup. We define one CEP Query there. Dummy BPEL process does nothing but subscribes to the WS Resource that is created by CEP service for this Query and has some dummy operation for notification processing.

7. Conclusion

We find that WSRF standard is a general and convenient way for information exchange between different systems. WSRF support for notification allows representing events and messages and gives additional advantage of publishing of the structure of information. We demonstrate that it can be successfully used for a general integration mechanism between CEP Network and BPM Solution. In the very similar and straightforward way we could provide integration between BPEL process and e.g. BI tools.

We find that this integration solution can be easily used in SOA environment where ESB middleware is used for message routing. Since we use web service standard message exchange between CEP Service and BPEL Engine can go through ESB as well.

We also find CEP Network quite useful abstraction which can be used when considering integration with multiple CEP Engines.

References

- [1] Complex Event Processing, white paper, IBM, 2007, <http://domino.watson.ibm.com/comm/research.nsf/pages/r.datamgmt.innovation.cep.html>
- [2] Web Services Description Language (WSDL) 1.1., W3C, 2001.; <http://www.w3.org/TR/wsdl>
- [3] Web Services Business Process Execution Language 2.0, OASIS, 2007.; <http://docs.oasis-open.org/wsbpel/2.0/>
- [4] Web Services Notification 1.3., OASIS, 2006.; http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn
- [5] Web Services Resource Framework 1.2., OASIS, 2006.; http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

²⁶ <http://ode.apache.org/>

²⁷ <http://www.eclipse.org/helios/>

- [6] Tammo van Lessen et al.: A Management Framework for WS-BPEL, ecows, pp.187-196, 2008 Sixth European Conference on Web Services, 2008.
- [7] Esper Reference Documentation 2.3, Esper, 2009, http://esper.codehaus.org/esper-2.3.0/doc/reference/en/pdf/esper_reference.pdf
- [8] S. Zdonik et. al.: *Towards a streaming SQL standard*, PVLDB 1(2): 1379-1390, 2008.
- [9] D.Luckham: *A Brief Overview of the Concepts of CEP*. <http://complexevents.com/?p=399>
- [10] G Sharon, O Etzion, Event-processing network model and implementation, *IBM Journal of Research*, V 47(2), ISSN 0018-8670, 321-334, 2008
- [11] Open ESB Intelligent Event Processor, <https://open-esb.dev.java.net/IEPSE.html>

This page intentionally left blank

Business Information Systems and Software Process Improvement

This page intentionally left blank

Value and Viability Considerations in Information Systems Development

Einar POLIS

Tallinn University of Technology

Abstract. The question of value is being increasingly discussed in various disciplines in recent years. Value-Based Management and Value-Based Software Engineering (VBSE) have emerged as promising attempts for employing the value concept in generating information systems which satisfy the needs of all stakeholders. We analyzed several package software implementation projects for assessing applicability of value based approach and found that seldom sufficient value for stakeholders has been produced. VBSE might have helped to achieve better results, although the theory itself is not fully matured yet. We found that stressing the role of agency in value analysis by utilizing the life metaphor can significantly enhance the explanatory power of the theory. This facilitates understanding of many advanced qualities of systems and development of theoretical framework for transforming the enterprise together with its information system. Our analysis will be concluded with suggestions for conducting projects similar to those analyzed in this article

Keywords. Information systems, value-based software engineering, enterprise modeling, living enterprise, self-organization

Introduction

Some management theorists and practitioners [1;2] have studied long-term success of enterprises and found useful to compare enterprises with living organisms. Life metaphor is frequently used by other kind of system builders too (“system has gone live”, “deadline”). Nevertheless, for a more rigorous approach, elaborated theories are needed. To explain the nature of living systems, for years have been around theories like Living Systems Theory [3] or Viable Systems Model [4]. Building architect Christopher Alexander [5;6] has written extensively about living structures and has already influenced information systems domain heavily by introduction of the design pattern concept [7], though his latest ideas have not gained widespread following yet.

Development of information systems has frequently to deal with great complexity in a limited timeframe. It is commonly considered to be a social and creative activity, which is difficult to formalize. Appropriate mindset and constructive worldview are needed for achieving higher success rate of projects and for bringing more value to stakeholders. We claim that organizational analysis using metaphors like living enterprise is beneficial at least for certain information systems development projects, especially in choosing an appropriate development strategy. However, despite the potential, only few attempts (e.g. [8]) have been made for moving from generic concepts closer to the level of details necessary in practical development work.

We are trying to approach this issue from a different angle. Value-Based Management (VBM) and Value-Based Software Engineering (VBSE) have emerged in recent years as alternative ways of organizing enterprise change management and information systems development respectively. We used their tools for analyzing several core banking systems replacement projects. Although such tools appeared to be useful for explaining project outcomes, at their current state the methods are lacking ways for dealing with some aspects of systems viability, therefore we would like to propose a complementary approach to value analysis.

1. Problem Statement

The issues tackled in this article are originated from relatively common packaged software implementation projects. In the last decade the author has been dealing with implementation of core banking software from the leading software vendor in several financial institutions, with variable rate of success. Main characteristics of in total of 13 projects are summarized in Table 1. Individual projects are referenced with letters A to M throughout the article.

Table 1. Summary of sample projects

Characteristic	Scale	Explanation (Occurrences)
Management style	Autocratic	Decisions made by few individuals, only minimal amount of information communicated (D, E, F, G, J, L)
	Democratic	Most decisions widely discussed and communicated (A, B, C, I)
	Self-organizing	Experienced people tried to find the way out (H, M)
Communication style	Informal	Significant amount of time spent to networking, ad-hoc discussions (E, F, G, H, J, K, M)
	Procedural	Documented processes, recurring meetings (A, B, C, D, I, L)
Customization level	Low	Alignment to product, new/limited implementation (A, J, K)
	High	100k or more lines of new code, custom modules, etc. to align with business requirements (B, C, D, E, F, G, H, I, L, M)
Business/technology balance	Business	Total dominance of business reasoning (A, B, C, D, E, G, H, I, J, L, M)
	Technology	Technological arguments were at least listened (F, K)
Sourcing	In-house	Mostly conducted by local staff (-) and partners (A, B, C)
	Outsourced	Mostly (D, E, F, G, I, K, L, M) or almost fully (H, J) staffed from external sources
Resources	Scarce	Limitations in staffing, ad hoc cost-cutting efforts (C, I, K)
	Plentiful	Resourcing had minor impact (A, B, D, E, F, G, H, J, L, M)

There are many similarities between the projects, which makes them comparable and suitable for investigation. Related institutions were mostly small to medium size branches of large multinational corporations. Though some parts might have been outsourced to cost-efficient countries, majority of work was conducted in developed countries of Europe, Asia, or Middle East, by mixed teams of local staff and consultants from international agencies. Of course, there are also some factors we had

to ignore for simplifying the analysis, like regional cultural differences or global economic situation at the time of conducting the projects.

Serious problems were encountered in all projects. Even when the usual success criteria were more or less met, high stress level caused a lot of suffering for stakeholders (B, K). Most projects were preceded or were expected to succeed by similar projects in other branches of the bank (except G, J), but only in few instances significant reuse of customizations and noticeable learning from previous experiences took place.

Poor result of projects had serious impact to organizations, which were unable to function without efficient information systems and where implemented software system was expected to possess a central role. However, we believe that encountered problems are not specific only to those institutions, but are variations of classic complex questions from enterprise information systems development domain: Why so many information system projects fail? Why experienced developers and enterprises are not learning from failures? How to measure the success of enterprise/information systems designers?

2. Value Analysis

There are many possible reasons for failure of sample projects. Unrealistic goals, budget size, poorly managed risks (bad luck), availability of knowledgeable and motivated hardworking people are all important success factors. In some cases these issues can be tackled. In better performing sample projects (A, B) from the time/functionality/costs point of view, wide range and complex functionality was introduced while leaving the practical implementation work to modestly paid developers, who learned specifics of a particular software system by doing and also whose banking knowledge level was usually below the financially significantly better rewarded external consultants of other projects.

Similar problems with information systems development projects in general are frequently reported in literature. This leads to assumption that inappropriate understanding of value is quite common in projects and can possibly trigger abovementioned undesirable consequences. From the other hand, if personal and organizational values are aligned and appropriate, high-quality information systems can be generated more likely.

Value-Based Software Engineering (VBSE) is an emerging theory originated mostly from the work by Barry Boehm [9]. By definition, its aim is to connect traditionally value-neutral computer science with value-based theories, such as utility theory, decision theory, dependency theory, and control theory. Central claim of VBSE is a theory W, also known as the Enterprise Success Theorem, which states that “Your enterprise will succeed if and only if it makes winners of your success-critical stakeholders”.

Financial value has central position in VBSE and it is no surprise that concentration on financial value is common for financial systems and institutions too. In some cases (like with project costs) it is quite well measurable and in some cases where it is not (like ROI), it is still comprehensible. Various tools have been proposed for performing actual value analysis in VBSE, though no method has become prevalent yet. Therefore we are using relatively informal approach for identifying stakeholder value satisfaction in sample projects. The results of this activity are presented in Table

5, with separation of traditional, mostly financial values and alternative values discussed below.

Table 2. Stakeholder roles

Role	Bank	Supplier	Description
Owner	Usually large public or state-owned institution	Usually a small company owned by few people	Represented through management hierarchy, which can influence projects by financing, appointing key persons, steering the progress and conformance with strategic objectives. The ultimate source of organizational culture
PM(O)	Project manager or managers, sometimes accompanied with administrative workers (PMO)	Usually more like a resource manager, sometimes team leader	Coordinates different activities or work-streams
Business Analyst	Business representatives and/or system analysts	Functional consultants, mostly with banking, sometimes technical background	Identifies business requirements, usually coordinates development activities in particular area. Usually no drastic changes to business are caused by such projects
Architect	Sometimes sets standards, requirements to interfaces, etc.	Usually just a title for praising outstanding developers or for lead developer	Should define high level structures, interoperability with other systems, development standards
Developer	Usually new to the software system, sometimes not present	Product experts from technical side, knowledgeable on software specifics	Complexity level of developments, sophistication of tools, etc. is not particularly high, which frequently causes unjustified underestimation of the developers role
Tester	Business Analysts in smaller, dedicated testers in bigger projects	Sometimes fully outsourced to separate company	Should ensure reasonable quality of the system
Infra-structure	System administrators	Configuration manager	Provides environments for growing the system. Usually quite transparent, but in case of failure affects everybody
User	Mostly noticeable in requirements elicitation, UAT (User Acceptance Testing) phases of the project and from feedback from the usage of the deployed system	Theoretically can be the bank's customers, but this relationship rarely matters	The people who are going to utilize the system in performing their business activities
Customer	Could be direct users of the system via electronic channels	Meta-customer for the supplier, but nevertheless important	Entities to whom the enterprise has to provide value in order to survive

We simplified the roles of stakeholders by removing the details which were irrelevant to this kind of analysis, in order to make projects comparable and complexity manageable. Results of this effort are presented in Table 2. As all the projects were conducted with the help of third party companies, a distinction is made on how the roles are carried out by different organizations. In larger projects the roles were split into more specialized roles or assigned to teams. Alternatively in smaller projects one person could perform many if not most of the roles. Though some roles may seem more important than others, it is difficult to point out any roles which are not success-critical. Some people are also usually considered more valuable or important for the project, which is reflected in their appropriate (principal, senior, etc.) titles, in managerial hierarchies and ultimately in rates. Desire to keep the latter down has huge impact on outcome of the projects in globalized economy, because of opportunities and temptations for staffing or outsourcing parts of it into low-cost countries.

The analysis led to several conclusions. Firstly, too heavy or too light emphasis on financial value could hurt the projects significantly. There are also less obvious findings, like domination of business analysts (B, D, G) or accountants (special kind of representatives of owners' interests) over developers can jeopardize project success (D, E) completely. Attempts to go live as fast as possible usually led to better results, though when was done in administrative way (F) or with too little of testing (J), caused serious problems later. Other success factors were strong local involvement, efforts in keeping system integrity and preparedness to change of requirements.

Most important success factor seems to be a personal attachment to the outcome. This surprisingly does not have to be reflected in huge amount of overtime hours, but more on whether the people really care about the project and the organization. If we feel connection with something, we most likely wish it to last, therefore the enterprise must possess some characteristics of the living entity (desire that the project lasts indefinitely is hardly justified). Indeed, even if financial aspects remain most important at least till a certain extent [10], it is difficult feel deep connectedness with money-making machines or body-shopping companies. Softer aspects are hard to tackle in a reductionist way, and there is a frequently discussed gap between objective and subjective worldviews today [11]. Nevertheless, we suggest that not taking into account viability of the enterprise is causing serious problems for many information system development efforts. This leads to outcomes which are not able to survive in changing environment, if they become into existence at all.

3. Viability Analysis

Systems development as an intellectual and creative activity depends a lot on the way of perceiving the world. According to the interpretive paradigm [12], enterprises have tendency to become similar to what we are thinking of them. For example, if we consider them living systems, they are more likely to possess qualities usually attributable to living organisms. We believe that not only the senior managers have the power to influence the enterprise with their thinking, but all stakeholders.

Some management theorists and practitioners have studied longevity of enterprises and found many similarities between them and living systems [2]. Others have found the life metaphor just useful for explaining the functioning of enterprises [1], especially when they behave like intelligent or learning entities. It is meaningful to say that

certain characteristics make the enterprise more alive and others less, like presented in Table 3.

Staying alive is universally important for individual beings, whose ultimate goals are frequently thought to be survival and replication, or long life and prospering. Opinions differ on necessity of dying, its importance in evolutionary adaptations and in adding value to lives of humans and enterprises [13]. Disappearance may cause significant economical damage, loss of knowledge and human suffering. From the other hand, entities might have moral reasons for restricting the freedom in achieving their goals. These are based on speculations (like free will, common good, or consciousness) on the nature of the world, which are unavoidable in composing the practical action plan. System builders should consider these questions, though likelihood of finding a consensus is probably small.

Table 3. Comparison of characteristics of the enterprise

Characteristic	Less Life	More Life
Organization	Allopoietic, structure is produced outside	Autopoietic, self-organization
Control	Centralized, autocratic	Decentralized, distributed, democratic
Optimization	Concentration on core processes and values	Diversification
Identity	Anonymous, interchangeable	Distinguishable, personality
Utility	Serving somebody's interests	Exists primarily for its own survival
Ownership	Changing, disrespectful	Tolerable, nonintrusive
Attitude	Indifference	Connectedness

Social systems can only be sensed indirectly and are more dependent on our way of thinking than engineered systems. Nevertheless, similar qualities or architectural principles characterize good design of information systems (entropy resistance, growth accommodation, etc. [14]) or buildings [15]. Regarding the latter, Alexander [5] has comprehensive discussion on difficulties in finding appropriate words for describing such qualities. Freedom, wholeness, and quality without a name are all closely related to liveliness. Opposite characteristics are attributable to lifeless, machine-like structures. Realistically our creations contain elements both for increasing and decreasing liveliness, thus making the whole more or less alive. Moreover, sometimes we may want to have a machine-like tool, shelter or enterprise, only to produce concrete results in a particular timeframe. Universality, self-organization, etc. characteristics in that case are only hurting the efficiency.

Ability to distinguish living from non-living aspects seems natural to human mind, thus the life metaphor has a good potential to be well understood by different stakeholders of enterprise and it is also a good candidate for facilitating understanding and concentration on important things in information systems development. Relative popularity of life sciences at present can be beneficial too. However, finding precise definitions for concepts in this domain is relatively challenging and is still ongoing activity. Existing theories are mostly intended for senior managers of companies, thus lacking many details of practical implementation.

Life is an emergent property of complex structures, which are usually having characteristics like metabolism, growth, reproduction, adaptation, homeostasis, autopoiesis, etc. Several generic theories have been created for describing such systems,

including Viable Systems Model (VSM) [4] or Living Systems Theory (LST) [3]. The former finds 5 subsystems necessary to meet the demands of surviving in the changing environment for any autonomous system. The latter presents a hierarchy of living systems and identifies 20 subsystems for processing matter/energy or information present in them all. From enterprise information systems development point of view, specialized theories could be even more applicable, like Luhmann's theory of autopoietic social systems [16] and especially Alexander's approach [6].

Alexander has asked many people on liveliness of different objects and found much communality in their opinions (such experiments are of course dependent on presentation of objects and his books use images from renowned photographers, certainly capable of projecting desired qualities). He also collected supportive examples from wide variety of natural phenomena and built comprehensive theory for explaining this. His basic element for explaining living structures is centre. By this theory we should treat the environment as a field of wholes, each supporting and amplifying one another. One can be very precise and descriptive about these wholes, but cannot avoid looking at the totality at each step of the way. Centers are not isolated entities, but work together for supporting life and consist of other centers.

Table 4. Alexandrian properties of living structure with references to systems implementation (excerpt)

Property	Narrative	Implementation
LEVELS OF SCALE	Within or around given centre, exist smaller centers which are level of scale lesser in size	System must be divided into reasonable number of modules/subsystems (e.g. level of scale or 7 ± 2)
STRONG CENTERS	Centers have been reinforced by other centers and intensified by STP	System must have clearly identifiable non-competing concentration points
BOUNDARIES	Strong boundaries enclose the centre and adjacent centers around	Autopoietic systems are operationally closed, but interactionally open and structurally coupled with environment
ALTERNATING REPETITION	Recurring structures throughout the design in non-trivial way	Basic elements should be reused in several places with minor modifications only
POSITIVE SPACE	The region of the design filled with latent centers	Not only what is designed has importance, but also what is intentionally not designed
GOOD SHAPE	As centers become reinforced and intensified, latent centers become definite centers	Obscure structures should be avoided, if there is no absolute need for using them
...
NON-SEPARATENESS	As the structure develops through its uncompleted forms, exaggerated differences are eliminated	The system has to become maximally adapted to its surroundings

Through years of careful analysis he identified around fifteen structural properties which are constantly present in living systems. These are illustrated in Table 4, with suggestions for implementation in transforming the enterprise. Stereotypical centre configurations along with these properties are special cases, earlier referred as design patterns [7]. By Alexander the patterns were expected to express collective experience, not so much based on formal proof on why they are working, but on common sense and shared vocabulary. Design patterns are used extensively in many domains: especially in object-oriented programming and other disciplines of software engineering like

systems analysis, data modeling, software configuration management, project management, business or organizational modeling, and teaching. Several big software houses have published papers with patterns for using their products. Unfortunately most pattern implementations have concentrated on utilitarian value of the individual pattern as “a solution to a problem in a context”, by omitting most of Alexander’s philosophy, generative power of pattern languages and thus possibility for explaining how true masters are working. This might be one of the reasons, why design patterns are criticized in recent times by acclaimed programmers [17].

Surroundings and organizational culture are important building blocks of the living enterprise. This helps to explain the raise of design patterns movement and why they may be going out of fashion. It is still uncertain if we are culturally ready to grasp most recent ideas of Alexander [6], which are even more unorthodox than the thinking which led to widespread recognition of design patterns in software development. Nevertheless, they can provide a systematic approach for transforming the enterprise together with its information system. Structure-preserving transformations (STP) are interpretations of structural properties from Table 4 applied sequentially for increasing the liveliness of the system. They facilitate accommodation of growth from basic structures into complex living entities, by maintaining the system integrity and directing it towards constantly elaborating requirements. This correlates well with approaches promoted by many modern agile software development methodologies.

4. Information System of the Living Enterprise

Information and system are broad concepts which are difficult to define, similarly to the concept of life. To have a practical and tangible interpretation for information system of the living enterprise, we treat it as a product of externalization of data the entity has about other entities or things in general. By externalization we mean outputting facts, assumptions, goals and other relevant items from minds to external (computerized) system, instead of relying only on direct relationships between them. The idea of externalization is inspired by GTD method of personal productivity expert David Allen, whose experiences have shown that freeing the minds by externalization increases both productivity and happiness [18].

The needs of enterprises as living entities are somewhat similar at the high level of abstraction and therefore externalization of their information processing work should produce similar structures. They all need to memorize the data relevant to their existence and to have means for generating new information or improvise based on that data. So far we have been significantly better in externalization of storing and retrieval of the data, than automated knowledge creation. This finding is reflected in the structure presented in Figure 1 as a generic semantic architecture of the information system of the living enterprise. All enterprises should have an action plan (strategy, projects, tasks, procedures, measurements, etc.) identified and assets tracked, mostly for internal purposes (“Personal References”). Some data (transactions, services) is created for sharing with other entities (“Public References”). There might also be a need for organizing or even producing and sharing reference materials for general public (“Media library”). Historical data (old transaction data, communication logs, etc.) is also frequently useful (if not required legally) to keep.

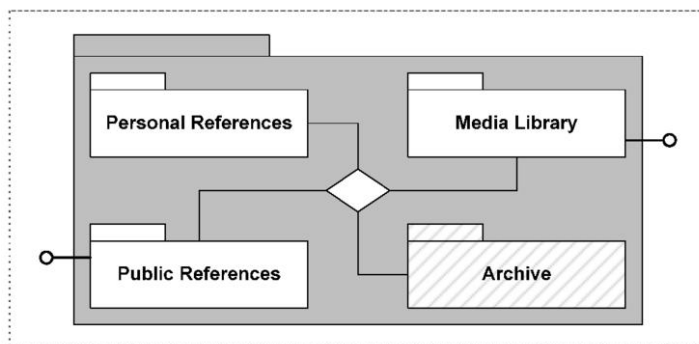


Figure 1. Semantic architecture of the information system

Technologies for storing enterprise data, including e-memories of their employees, are advancing fast [19]. Optimistic futurists also predict that in upcoming decades the machine intelligence or at least computing power will surpass the human brain [20]. Information systems empowered with such abilities will be certainly very different from the systems we can build today, but we should have such transitions in mind while developing our present systems. In sample organizations, however, data collection and practices were quite far from earlier presented structure. There has been a struggle even in capturing legally and operationally vital data (C, M); organizing professionally important media was attempted only in biggest organization (L) and even there with moderate success.

Enterprises and their information systems form complex hierarchical and networked structures, with overlapping sections and fuzzy borders. Basic building blocks of such structures are communications between individual persons, which give arise to different enterprises and to the system we can collectively address as the society. Figure 2 is a simplified presentation of such structure, comprising from just two persons, two enterprises and their information systems. As we may see, it is even with small number of participants difficult to capture all the relations and even more, to present them graphically.

Different aspects of liveliness affect information systems design at micro-, macro- and ecosystem levels. In general, machines are expected to relieve us from mechanistic repetitive activities, including in information processing. People and teams should be left with freedom of choice, creative problems, and possibility to grow with the system, which with respective rewarding feeling can make them happier [21]. Properties of things themselves influence our feelings too, we may intentionally want to make information systems look better for increasing their chances for a longer life and making their users feel better. Although beauty of buildings and many other human creations is extensively discussed, beauty in information systems design is generally considered unimportant, possibly omitting an important factor in analyzing its longevity. There is a related discussion about designing a clock to last for 10 000 years, where important objective is making it appealing to people, for increasing chances for survival through all the future uncertainties [22].

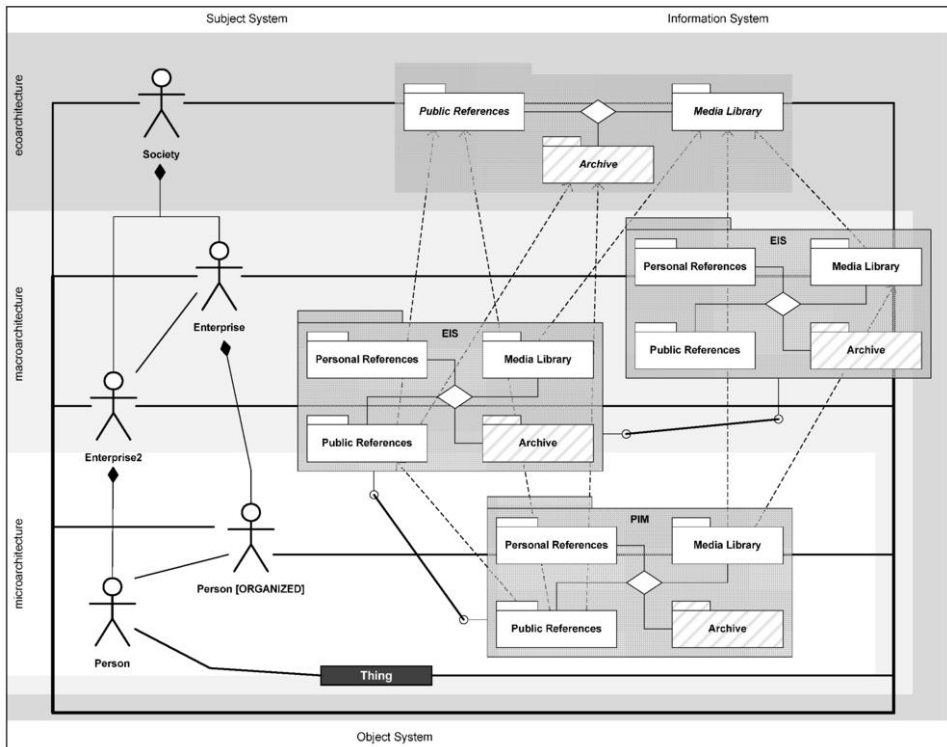


Figure 2. Information systems of living entities

As with people, the enterprises or their organizational units may have different level of organization. In well organized enterprise its rules and policies, goals and values, actions for achieving them and data on important things are externalized into enterprise information system (EIS). Among other reasons, this makes its pattern less vulnerable to personnel turnover. From the other hand, we may expect that realistically not everybody is willing to share and externalize her specific know-how. Likewise there are always people who are reluctant to get organized and prefer to keep the system of organization in their heads.

The society is commonly considered to be self-organizing life-like structure. Governed by nobody, nevertheless it provides us with rules to follow and even with some public data. From the perspective information systems is important to understand the viability of platforms and standards, and to adjust development activities accordingly.

Information systems of sample organizations are hardly the result of carefully planned conscious efforts. Only in few democratic organizations (C, I) preferences of individuals other than high-level managers were taken into account, if they were not presented as corporate requirements. Only in few cases the environment was considered stable and deterministic (D, H). Nevertheless, long-term planning and accordance with societal processes were never extensively discussed.

5. Coevolution of Enterprise and Its Information System

Life metaphor can be useful for analyzing both enterprises and their information systems. However, due to their specifics we should treat them as different coevolving species. We can only transform them separately, although depending on each other and from the speciesist viewpoint, more in favour of the enterprise. Among other possibilities, this can be used for analyzing relationships between so called business and IT people. Optimal levels are difficult to achieve, former stereotypically tend to be better communicators, but are usually unable to decide technology questions.

To be alive is a process of going through different stages of life, in each having different levels of liveliness and relations with the environment. Lifelines of the enterprise, its information system with subsystems, and development organization are illustrated in Figure 3. We could have divided a lifeline into more stages (e.g. youth, sickness, prospering, decomposition) or expressed hope that some entities might be eternal. Information systems are expected to cease to exist as the enterprise discontinues, though common thinking is mostly concerned about the life of separate subsystems, which are likely to be decommissioned and replaced with new systems from time to time. Sometimes the subsystems begin their independent life as spin-off packaged software, as in case of software implemented in sample projects.

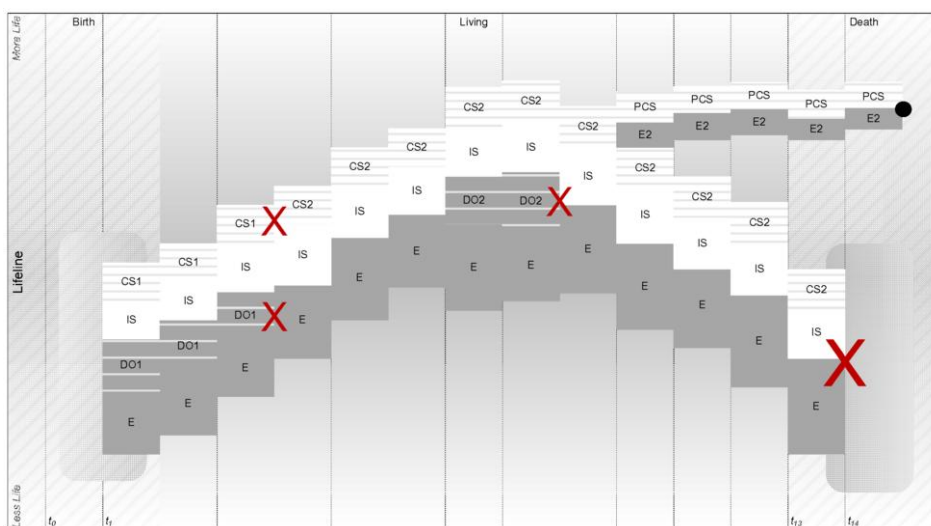


Figure 3. The lifelines of the enterprise and its information system

We are mostly interested in directed coevolution. Structure preserving transformations provide suitable framework for gradually changing the enterprise and its information system in desired direction, usually towards increased liveliness. Transformation steps should bring us closer to this and to related generic goals of the enterprise. We can consider activities like sample projects such steps too, though complex and targeting several subgoals at once. Their aims in practice were stated differently though, like increasing revenue, cutting costs or being able to extend the product range. Combined with other traditional measurements like personnel turnover, market share or public image, their concrete values are also important components of combined indicator of liveliness, in addition to characteristics discussed earlier.

In practice our transformational activities tend to be focused on bigger changes, organized around projects and development programmes. After achieving related higher-level goals, we should find the enterprise in the next stable state. Goals are converted into requirements for information system (or software) development process, usually with more concrete kind of organization. The latter roughly assumes either predictable or chaotic environment, proposing correspondingly either machinelike waterfall or more flexible agile approaches, without much analysis like proposed here for identifying the actual situation.

To deal successfully with demanding tasks, development organizations have to involve experts from various disciplines. They all can provide valuable insights on volatility of ecosystem, politics inside the company or technology trends. Experiences of growing together with the enterprise count in doing that. This is often expressed by strong corporate culture, which ensures that the enterprise has not lost its identity after possible in- and outsourcing activities, mergers and acquisitions. Culture can be supported by appropriate artifacts, including tools for transformation. In best cases custom standards and proven infrastructure for change will emerge. Also information systems of development organizations can be partially retained from one organization to another, so that learning from previous experiences will be externalized.

Brand describes how buildings change over time and adapt to their surroundings [15]. Likewise some components are more suitable for creating living and lasting enterprise information systems. Information systems have to adapt to change at even greater pace as computer hardware and software becomes obsolete fast. Open source and homebrew software with personal attachment may have better chances for survival, reuse, and refactoring. Data must be kept alive by choosing open widespread formats if possible, upgrading them on necessity and ensuring it has value for the users. Contracts should leave freedom for adapting to changing environment and constantly improving understanding of the system.

Some experienced consultants have quite accurately predicted the outcome of sample projects, without much formal analysis. This should make us optimistic about possibilities for adding predictability to such efforts. So far seems that development methods are chosen based on personal preferences or company traditions, not based on suitability. Neither traditional nor agile methods in sample projects were followed methodically. Imbalance between business and technology people caused problems in several projects. Only generic office software (Microsoft Office, Visio, Project) tools were used in basic level and bug/defect tracking software. Rarely non-software artifacts were reused in subsequent projects. Surprisingly core of the implemented packaged software has shown strong viability, with many lines of code older than 30 years.

6. Alternative Analysis Example

We have tried to complement VBSE with the insight that personal attachment and appropriate mindset are vital for projects, for long-term success of the enterprise and for each related individual. Table 5 in a next page attempts to present the results of this effort. Of course, there are many different aspects and different ways for representing the information. Moreover, we could fit here only a comparison of two quite opposite projects. Therefore we have to summarize main findings separately: in our project sample, better results were achieved in case of long-term partnership between institutions (as evolution needs time) and in an atmosphere of mutual respect and trust.

Table 5. Excerpts from projects/stakeholders/value analysis

#	Stakeholder	Value for Bank	Value for Supplier	Viability assessment
A	Owner	Possibility to expand local product offerings	Regional partnership at reasonable profit rate	Financially the projects performed well and also other objectives were met. Stable business environment was created, which was reused in subsequent project B.
	Project mgmt	One more project within the same company	Volume-based compensation	
	Business Analyst	Opportunity for self-development	-	
	Architect	-	-	However, after completion aggressive ROI period started, which led to break up of the development organization and partnership with the vendor. Support and future developments centralised to cost-efficient country, which made future developments and necessary software upgrade increasingly complicated
	Developer	Usually cheaper than using the supplier	Relatively good income for no more than 40h of work per week	
	Tester	BA-s understood the cost of errors	Unit testing and fixing usually chargeable work	
	Infra-structure	Less laborious maintenance and production support	-	
	Users	Dissatisfied with the product, but with no real reason	-	
	Customer	Attractive products and good rates were offered to them	-	
.....				
D	Owner	Possibility to expand local product offerings	Easy money before financial crisis	The project was stopped after two years, restarted in other location by repeating the same mistakes and without success. Business needs were changing and unclear, nevertheless formal waterfall-like approach was tried. Despite big financial losses, permanent staff was able to keep their jobs. External consultants were mostly satisfied with the project, many have found this time valuable part of their lives and formed social connections have survived over the years
	Project mgmt	Negative feelings compensated with money	Impressive title with little responsibility	
	Business Analyst	Learn something valuable in a job market	Good guaranteed income	
	Architect	Proposal of unrelated but professionally interesting services bus	1 week job to draw a later discarded diagram	
	Developer	Left because of low pay compared with supplier	Little professional growth, but pleasant atmosphere	
	Tester	-	Opportunity for third-party company	
	Infra-structure	Mostly one dissatisfied person who finally left	Extra responsibilities, but compensated	
	User	Effort for requirements presentation wasted	-	
	Customer	No better service as no new system	-	

7. Conclusions

The aim of this article was to demonstrate benefits of using the living enterprise metaphor from organizational analysis in information systems development process. Few similar attempts (e.g. [8]) have been made so far. However, we expect such approach to become more widespread, as complexity of business and technological systems is increasing.

Good metaphors are needed for consolidating viewpoints of stakeholders in information systems development. Life metaphor with its explanatory power is suitable for this purpose and also correlates well with modern systems development practice. From the other hand, living entities are generally more difficult to study than non-living things and experimentation with them is no less complicated. Uniqueness of every organism makes reproduction of experiments difficult. Nevertheless, despite of limited amount and quite informally studied sample projects, we hope that we were able to clearly present various aspects of the approach.

For connecting ideas from management theory with existing information system development theories, proposal for extending VBSE theory was made. Viability of the system can be indeed useful for identifying requirements, consolidating viewpoints and ensuring longevity. Additional work is needed for elaborating presented concepts and testing them in practice. As concept of life covers wide range aspects, some potentially useful building blocks for theory were left untouched here. Maybe not so much for practical use, but for building more sound groundings, presentation might have benefitted from more rigorous semiformal language. Having identified the importance of value and viability for enterprises, we would like to propose a modified Enterprise Success Theorem as “In a long term, enterprise succeeds if and only if it makes winners of success-critical stakeholders and maintains its viability”.

References

- [1] G. Morgan, *Images of Organization*. 2nd Revised Edition, Sage Publications, 1997
- [2] A. de Geus, *The Living Company*, Harvard Business School Press, 1997
- [3] J. G. Miller, J. G.: *Living Systems*. McGraw-Hill (1978)
- [4] S. Beer, *Brain of the Firm: the Managerial Cybernetics of Organization*, Wiley, 1981
- [5] C. Alexander, *The Timeless Way of Building*, Oxford University Press, 1979
- [6] C. Alexander, *The Nature of Order*, Center for Environmental Structure, Berkley, 2003
- [7] C. Alexander, *A Pattern Language*, Oxford University Press, 1977
- [8] D. McDavid, Systems Engineering for the Living Enterprise. *18th International Conference on Systems Engineering*, 16-18 Aug. 2005 (2005), 244–249
- [9] S. Biffl, A. Aurum, B. Boehm, et al. (eds.), *Value-Based Software Engineering*, Springer, 2006
- [10] B. Frey, *Happiness: A Revolution in Economics (Munich Lectures Series)*, MIT Press, 2008
- [11] S. Kaufmann, *Reinventing the Sacred*, Basic Books, 2008
- [12] J. Dietz, *Enterprise Ontology: Theory and Methodology*, Springer, 2006
- [13] A. de Grey, *Ending Aging*, St. Martin's Press, New York, 2007
- [14] D. Spinellis, G. Gousios (eds.), *Beautiful Architecture*, O'Reilly Media, 2009
- [15] S. Brand, *How Buildings Learn: What Happens After They're Built*, Penguin Books, 1994
- [16] N. Luhmann, *Social Systems*, Stanford University Press, 1995
- [17] P. Seibel, *Coders at Work*, Apress, 2009
- [18] D. Allen, *Getting Things Done: The Art of Stress-Free Productivity*, Viking Penguin, 2001
- [19] G. Bell, J. Gemmell, *Total Recall*, Dutton Adult (2009)
- [20] R. Kurzweil, *Singularity Is Near: When Humans Transcend Biology*, Viking Penguin, 2005
- [21] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, HarperCollins, 1991
- [22] S. Brand, *Clock Of The Long Now: Time And Responsibility*, Basic Books, 2000

L.O.S.T Records: The Consequence of Inadequate Recordkeeping Strategies

Erik A.M BORGLUND^{a,1} Karen ANDERSON^a
^aMid Sweden University

Abstract. Recordkeeping is about “making and maintaining complete, accurate and reliable evidence of business transactions in the form of recorded information.” Yet the research on electronic record management has focused on either electronic records management systems or long-term preservation. Little has been done that takes an holistic view of the entire continuum and the connection between current records management and preservation for future accessibility. In this paper we aim to further elaborate the challenges and effects that are the results when an holistic view of the entire continuum and the connection between current records management and preservation for future accessibility is not taken. The research has been carried out as a qualitative case study in a large railway infrastructure construction project, Ådalsbanan, where many records are operational for more than 100 years. We present results from this case study where the long-term recordkeeping strategy only succeeded to preserve records in form of documents, after that the project phase has ended. The records born in databases and in other business information systems were lost, for future use, when the Ådalsbanan moved from being a project to being a operational railway, that needs to be managed for at least 100 years.

Keywords. Digital preservation, Electronic recordkeeping, EDRMS, Records Continuum Model, Ådalsbanan

Introduction

This research is about recordkeeping, an area that has gained limited interest from the IS field. Recordkeeping is defined as “making and maintaining complete, accurate and reliable evidence of business transactions in the form of recorded information.”[1]. The definition of the term recordkeeping includes “the design, establishment and operation of recordkeeping systems” [1].

Records are sometimes described as sources for knowledge within organizations [2-4] and can be defined as “institutional memory” [4]. Capturing, storing and sharing of knowledge can give an organization advantages in productivity and innovation [5].

Records are evidence of business actions and transactions and are kept to support current business and accountability. Records need to be managed in such a way that their authenticity and reliability is protected if their evidential value is to be kept [6, 7].

Since the early 1990s when it became obvious that electronic records needed new management techniques [8], archival science has struggled to establish a basis for sound recordkeeping. To meet the new need for managing born-digital records [9-13],

¹ Corresponding author: Erik A.M Borglund, Mid Sweden University, 871 88 Härnösand, Sweden;
Email: erik.borglund@miun.se,

many organizations have implemented electronic records management systems (ERMS), and electronic documents and records management system (EDRMS) and do, and the effect of those implementations has gained interest from researchers. A study of the major scientific journals in the area of electronic recordkeeping, the Records Management Journal, and Archival Science, found many success stories about the positive impact organizations can gain by implementing electronic records management systems [12-15] Implementation of an ERMS and EDRMS is about change in organizations [10], and it is a long process [16].

In the last 15 years several standards have been developed in the domain of electronic records management. In *MoReq2*, the *Model Requirements for the Management of Electronic Records* [17], the functional requirements for electronic records management systems are described. *MoReq2* is sponsored by the European Commission as being necessary to underpin transparent transnational e-services in Europe. In the United States of America the Department of Defense has a design criteria standard for electronic records management software [18]. Software vendors in the United States of America that have electronic records management software must be DoD-certified to be able to sell their software to public agency customers. The International Council on Archives has also developed a set of *Principles and Functional Requirements for Records in Electronic Office Environments*. [19-21].

The archival community has focused separately on ERMS and EDRMS implementation and on long-term preservation. Little has been done that takes an holistic view of the entire continuum and the connection between current records management and preservation for future accessibility. Electronic recordkeeping needs to adopt a proactive approach to be able to fulfill recordkeeping requirements, and to enable long-term preservation [22]. Proactivity in recordkeeping implies that information systems involved in recordkeeping should be designed to fulfill recordkeeping requirements. That is, the successful capture, management and long-term preservation of digital records depend on appropriate design of the systems in which they are created and kept. Well-designed and carefully executed strategies for migration of the systems and the records they contain are also crucial for success. Attempts to preserve records as items, or individual digital objects, cannot preserve the context, which is essential to preserving the evidentiality, the integrity and the authenticity of records.

Long-term preservation of electronic records is a challenge, which has been argued for in both the archival literature [8, 23-27], and in the IS literature [28, 29].

In this paper we aim to further elaborate the challenges and effects that result when an holistic view of the entire continuum and the connection between current records management and preservation for future accessibility is not taken. We have carried out this research in a large infrastructure project of a railway construction, where many records are operational for more than 100 years, but current records management was separated from long term preservation strategies.

The remainder of the paper is as follows: first we present the applied research method and present the context in which the research has been carried out. Then we present the findings of the case study and an analysis of standards and methods for proactive design of electronic record management systems. The results are in the next section, which is followed by a discussion of the results. The paper ends with a conclusion in which outlines for future research needs in this research domain are drawn.

1. Research Method and Research Context

This paper is based upon a one-year research project, which aimed to identify problems and potential research areas in a large railway infrastructure project. The research project was explorative and is best described as an interpretative case study. Interpretive case studies are a commonly applied research method in information system research [30]. We have used interviews and official documents as the main data collection techniques. This research conforms to the established Scandinavian IS research tradition, with an intertwined mix of technologies, application areas, and stakeholders [31].

The research was carried out in the *Ådalsbanan Project* that is managed by the Swedish Rail Administration (SRA). The Swedish Rail Administration has overall responsibility for the railway transport system in Sweden. This includes the role of leading large infrastructure projects, which have an estimated lifespan of 150 years or more after construction is completed. The *Ådalsbanan* is a 180 kilometre-long new railway service between Sundsvall and Långsele in northern Sweden, which lies in the Bothnian Corridor: a strategically important part of the trans-European freight routes [32]. It is one of the largest railway projects in modern times in Sweden, and is also the first project of that size in which all records are born digital. The *Ådalsbanan Project* also involves several other official agencies: e.g the National Land Survey, the County Administrative Board and the Swedish Road Administration. Altogether there are more than 16 public organizations that are involved in the project. As well as the public organizations, hundreds of sub-contractors are involved in the project. The budget for the project is 6.6 billion Swedish kronor (approx €650 million). It takes a very long time to repay investment in new railways.

Large infrastructure projects in Sweden are commonly managed like the *Ådalsbanan* which is treated as a separate publicly-owned enterprise, with its own organizational boundaries, independent of the parent organization. In this case the parent organization is the Swedish Rail Administration.

2. Theoretical Concepts

Considerable work has been done on developing theoretical models that account for the entire recordkeeping environment and the activities and functions that take place within that environment. This section describes the basic concepts and the Records Continuum Model for an audience that may not be familiar with them.

Records and archives are the two concepts that are basic to archival science [33]. Two of the most widespread definitions of records are:

"Information created, received, and maintained as evidence and information by an organization or person, in pursuance of legal obligations or in transaction of business."[6]

"Recorded information in any form or medium, created or received and maintained, by an organization or person in the transaction of business or the conduct of affairs."[34]

The definitions of a record above are independent of format and do not make differentiate between electronic records and paper-based records. All records have content, structure and are created in a context [35]. The evidential value of a record is central, and records are preserved for the evidence they represent [4, 33, 36]. According to Cox [37] the evidential value of a record can only exist if the content, structure, and context are preserved. The context is the link between different records that belong together and also to the process in which the record was created. The record's relationship to transactions is both what makes records different from information and enables the evidential functionality of records [7]. In order to have evidential value, records must have two other criteria: authenticity, and reliability. A record can never serve as evidence if it is not reliable and authentic [6, 7, 25].

The Records Continuum Model was developed by Frank Upward, using Giddens' Structuration Theory[38] as a theoretical framework for its development. The aim of the model is to support archivists in their concern with the relationship between recordkeeping and accountability [39]. The term 'recordkeeping' used in the Records Continuum Model covers all aspects of management of records from strategic planning and design of systems through the many-faceted processes of capturing, managing, keeping and insuring access and usability of records.

According to Upward [39] a characteristic of the model is the view of records as unstable. A recordkeeping model should consider both an object-oriented approach and a system-based approach. There are no end products in an archival institution so there is a need for continuing addition of process metadata while they change across space-time. The model is four-dimensional (see Figure 1). According to Upward [39] traditional archival methods are creating one-dimensional documents and two-dimensional records, or three-dimensional archive but technologies enable a four-dimensional approach. Upward [40] writes that records can have multiple lives in space-time, and a record is never finished in its creation, it is continuously in change.

The transaction axis is about the core business, its processes and functions. The identity axis focuses on the information's relation to the organization and its actors. The evidence axis describes the status of the information, and what role it plays in the organization. In the recordkeeping container axis the focus is about what kind of aggregated information units the original record can gather later.

One way to describe the functionality embedded in the records continuum model is to relate the axis and their names to for example a business action, and the questions the model afterwards can help you to answer:

- Who (Identity) did what (Transactionality)?
- What evidence exists about this (Evidentiality)?
- What can be retrieved from the documents, records and archives (Recordkeeping containers)?

The model implies that a high qualitative recordkeeping needs to be able to answer above questions, the model also implies that when a record is created, it should already fulfill the requirements of the other dimensions. The model is the basis for the proactive approach where at creation a record should fulfill future requirements.

The Records Continuum Model also visualizes the relationship between records creation and the evidential value of records over time, i.e. between corporate knowledge and collective knowledge.

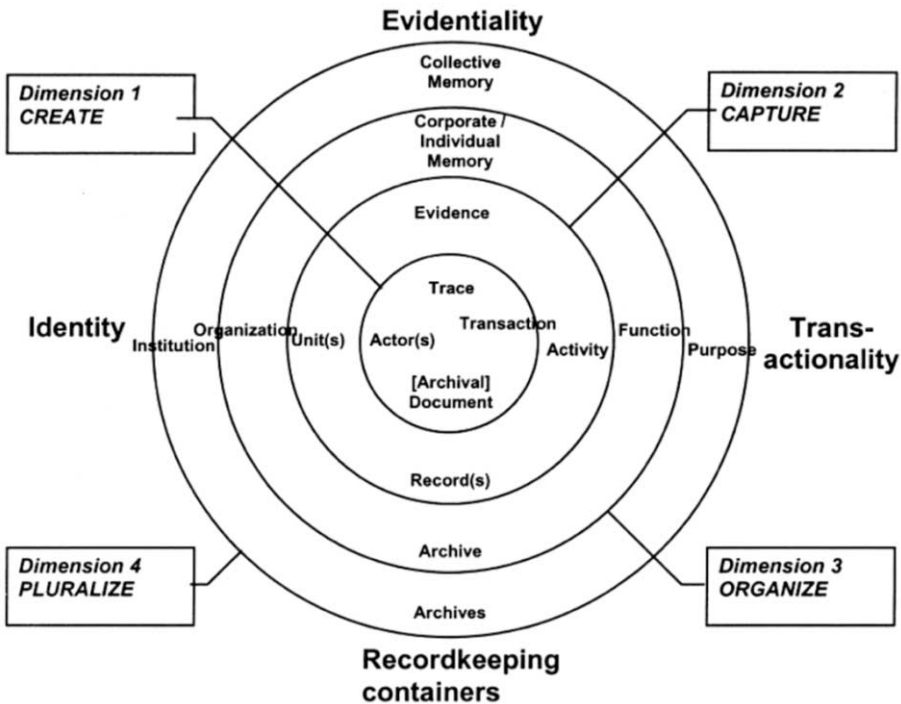


Figure 1. The Records Continuum Model[40]

3. The Management of Electronic Records in a Large Infrastructure Project

In Sweden all public records must be kept in such a way that the public have free access to the records. It also means that all public records should be preserved, if a decision for destruction is not taken.

The records management in this infrastructure railway project has been found to exist in three temporal structures. The notion of temporal structuring is suggested as a way to study and understand the role of time in organizations [41-43]. The three temporal structures are:

- The project phase, which includes all sub-phases within the project;
- The maintenance phase, which is the phase in which the railway is operational;
- The curation phase is the phase when the railway is no longer operational.

They are also seen in Figure 2.

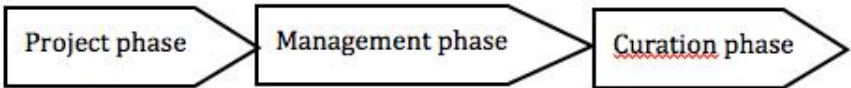


Figure 2. The three temporal structures of recordkeeping in the Ådalsbanan Project

Beside the records created within the project it is worth noting that many records are created within 16 other public organizations that have business that is affected by the railway project in some way. The full extent of how many records this is, and in what way it really affects the railway project in its three phases is unknown. No comprehensive cross-organisation records survey for the *Ådalsbanan Project* has ever been undertaken.

Within all the three phases above, the Swedish Rail Administration (henceforth named SRA) sets requirements for how the records in their electronic document and records management system (EDRMS) will be managed. They have applied a perspective that they believe covers the entire life cycle of the records, ie all three temporal phases. The SRA's aim is that their internal regulations will result in functionality that makes it possible to create, metadata tag, store, archive, sort out and preserve both paper based and electronic records for the long term.

The SRA has a distinct focus on the management of records that are in the form of a document in their official documentation. I.e. either the record is physically a paper document or they are electronic documents, such as Word documents, Excel spreadsheets, PDF files, or scanned documents such as TIFF or JPG pictures. All the records that are embedded in various business information systems are not covered by the SRA internal recordkeeping regulations. The consequence is that the records that are not in document format are uncontrolled. Therefore there is a very high risk that they cannot be preserved for the very long term required by this railway infrastructure project.

Empirical data from the project phase is presented below, but we also present how the two other temporal phases are organized and how they work.

The records belonging to the Swedish railway are geographically ordered, following the railway tracks. The majority of the records that are kept for the continuous maintenance of the railway are blueprints of buildings, the power supply, the signal devices, freight yards, and roads for which the SRA is responsible.

The records are kept physically in the regional SRA archive that is closest to the railway, and its tracks. Currently the records are paper-based in the SRA regional archives, and there is not yet any proposal for how to manage the digital records that result from *Ådalsbanan*. The SRA has chosen a solution for managing digital records whereby they preserve a clone (i.e. an exact copy) of the EDRMS, believing that in this way it will be possible to access all records even after the project is completed. However, there are many digital records in the *Ådalsbanan Project* that are 3D CAD blueprints that cannot be transformed to paper. There are detailed regulations about what records should be preserved during this temporal structure. The delivery of records follows a standardized process, where the project organization delivers the records to the management organization: that is, to the SRA operational division. Only the records that are necessary for the operational management of a railway are kept and accessible. Continuing fast access to these records is important, because if there is a problem with a railway, every minute that the train cannot run is costly.

All records that are not needed for the maintenance of the railway are assessed as to whether they should be kept or destroyed. Prior to this project the SRA has only needed to handle paper-based records. These records have been kept in the SRA's regional archives for a couple of years until the SRA delivered them to the Swedish

National Archives. The SRA has no defined and stated strategy for preservation of the *Ådalsbanan Project's* electronic records for the long term. The solution SRA has chosen for the long-term preservation of records that are not needed for railway maintenance is to make a clone, i.e. an exact copy of the EDRMS. At this point the SRA does not manage the records for maintenance and those for long-term preservation purposes separately.

3.1. Project Phase in Details

The *Ådalsbanan* railway construction project has a planned life of 10 years, and acts as an independent organization during this project phase. The project phase consist of three major sections: Investigation, Planning, and Construction (see Figure 3).

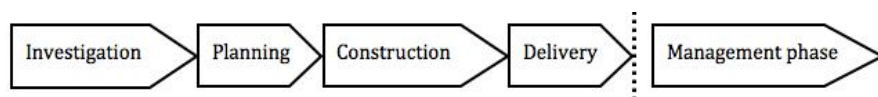


Figure 3. The project phases and its sub phases

There are well-described goals for electronic records management during the *Ådalsbanan* project phase. The following goals for record management are set out in the project description:

- The right information should be available at the right place, in time, with the correct status and fulfilling user requirements;
- The information and records management should fulfill the legal requirements for both public and secret records;
- All records should be traceable, and should be kept in the stated formats, follow stated standards, be managed in the stated way according to the SRA's and the project's internal regulations;
- All employees within the project and all internal and external partners should follow the stated regulations on records management in the project;
- The records that should be delivered for continuous management of the railway should follow the stated internal regulations for those records.

The *Ådalsbanan Project* has an EDRMS which contains more than 250,000 records, each of which can consist of many documents. The main purpose of the EDRMS was to guarantee that all records that should be available to the maintenance phase of the railway are managed correctly. The regulations set out what records and documents should be delivered for each part of the railway project and how that phase should be carried out. The EDRMS at *Ådalsbanan* is very simple: it looks like the file structure that you find in the explorer in a modern operating system. It is not structured to reflect and support the project's workflow. It acts more like a document repository, where the records are kept and preserved in a controlled environment. There is nothing in the regulations for records management in the *Ådalsbanan Project* about the importance of preserving context. This means that "Why-questions" about actions are

difficult to answer. There is no tradition within the SRA to make sure that the wider knowledge context attached to the records is preserved. SRA does preserve the transactional context, although they don't seem to have a strategy for preserving future metadata that should be captured if the records are used after the EDRMS is cloned.

4. Analysis of the Case Presented Above

The case study presented above indicates that the Swedish Railroad Administration focuses only on electronic records in the form of documents and on managing them in their EDRMS.

Electronic records management in the *Ådalsbanan Project* has three main purposes, which are:

1. To support the work carried out within the project, and during the construction of the railway;
2. To support the Swedish Rail Administration in its continuous management of all parts of the railroad during its' operational life;
3. To support the public's right of access to public records and future research requirements.

There is a clear separation between the strategies for the current records management, and the strategies for long-term preservation. In the current records management strategy the *Ådalsbanan Project* uses their EDRMS as the main repository for all created documents. However the strategy for long-term preservation focuses mainly on the preservation of the documents needed for the continuous management of the railroad, i.e. the second purpose above. These documents are e.g. blueprints, maps, and similar technical documentation.

We can critically state that many documents that are stored and managed within the EDRMS during the project phase are not necessarily included in the long-term strategy phase.

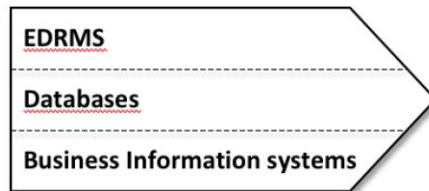


Figure 4. Existence of records outside the EDRMS

The Figure 4, should be understood as follows: The project phase includes records in form of documents that are managed within the EDRMS. However there are many records that are born digital and are managed within databases, and in various business information systems outside the EDRMS. The current strategy that the SRA applies, and that has been applied in the *Ådalsbanan Project* results in a situation where records created in databases and in other business information systems are not included in the long term preservation strategies. This result in a situation where many of the records created during the project phase will be lost for future use (see Figure 5):

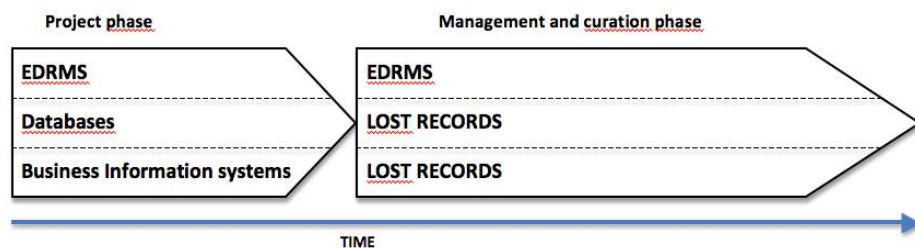


Figure 5. The LOST records

Figure 5 highlights that after the project phase has ended, the records imbedded in business information systems and in databases are no longer managed or preserved, i.e. they are lost.

5. Final Discussion

In the analysis of the records management in the *Ådalsbanan Project* presented above, we can see in Figure. 5 that the records created in databases and in business information systems during the project phase will be lost to the management and curation phases. It is of course easier to manage records that are documents, and can be treated as a unit. The records found in databases, and in other business information systems are less tangible. This results in a situation where the records born in the latter systems remain outside the long-term preservation strategy. The quality of the recordkeeping systems available to the management- and curation phases for the *Ådalsbanan* railway will necessarily be low, given the fact that many crucial records will no longer be accessible, because they have not been preserved.

As presented above, records and documents are sometimes described as sources of knowledge within organizations [2-4, 44] and can be defined as the memory of an organization [4]. Dimensions 3 & 4 of the Records Continuum Model signifies the organization and pluralization of records and their value so that they become individual, organizational and social memory. That is, the records are accessible to both the creators (individuals and the organization) and the wider society as evidence for accountability and research. Capturing the context surrounding the records is essential to such activities and implicitly underpins an organization's knowledge management. In sound recordkeeping practice it is important to capture the context in which every record is created, yet the EDRMS in the *Ådalsbanan Project* has a limited capacity to capture context. In archival science the central concepts of provenance and original order are about capturing and documenting the context and the relationship between records in a business [3]. The context is the link between records and transactions, processes and organizations and how they relate to each other. Without the provenance and the contextual links between records, records cannot be demonstrated to be authentic and reliable, evidentiality is lost and use of the records for knowledge and understanding about what has happened will be difficult. The SRA does not capture and preserve knowledge context and this makes it almost impossible for the records to act as sources for knowledge. This, together with the fact that too many records are not even preserved for the long-term will minimize corporate memory capacity enormously.

At the heart of the ISO 15489 standard [6, 45] lies the premise that records management and records management systems should support the business of an

organization. This realizes the benefits of sound recordkeeping, including the capture of corporate memory. The case study found that the *Ådalsbanan Project* EDRMS acted merely as a document repository and there was no evidence that work process analysis to identify where records are created had been undertaken. Thus the documents in the repository were not ordered in a way that reflected the business processes of the *Project*. By using a recordkeeping strategy that follows the work process the context is more naturally preserved, and the context is also implicitly embedded in the understanding of the process. A work process analysis also results in the identification of all kinds of records that are born in the business, even those that are born and managed in databases, and in business information systems. This proactivity is necessary for high quality recordkeeping. The Records Continuum Model [39, 40, 46] indicates that proactivity is essential for successful and complete electronic recordkeeping [47]. Proactivity implies that the following questions are answered at the systems design stage, before the records are created:

- What records exist in this business?
- In which systems are these records created?
- What records will be preserved?
- What metadata is needed at creation, and throughout the records' existence, in order to preserve context and ensure that the records remain evidential?
- How long should the records be preserved and what file formats are suitable preservation formats?
- What are the business needs for records' usability throughout their existence? i.e. long-term user requirements for the records.
- Who will be able to access and use the records in the future, and for what purpose?

A proactive work approach in the *Ådalsbanan Project* using a full work process analysis could have helped the SRA and *Ådalsbanan Project* to identify that many records are created and kept in their business information systems and in databases, in non- document formats.

6. Concluding Remarks

In this paper we aimed to further elaborate the challenges and consequential effects when an holistic view of the entire recordkeeping continuum and the connection between current records management and preservation for future accessibility is not taken. We have presented empirical results from a year-long case study of the large *Ådalsbanan* railroad project which provided a deeper understanding about the relationship between the long-term preservation problem and the operational focus and use of EDRMS. . The records created in the *Project* were planned to be managed in three phases: 1. The project phase; 2. The management phase; 3. The curation phase. We present results from this case study where the long-term recordkeeping strategy has only planned to preserve records in the form of documents, after that the project phase has ended. The records created in databases and in other business information systems will be lost to future users when the *Ådalsbanan* moves from being a construction project to an operational railway with operational needs for access to records from the construction phase for management and maintenance purposes for at least 100 years.

In this research we can conclude that the long-term preservation of records is very difficult, if not impossible, to achieve if the business does not proactively take this aim into consideration at the systems design stage. A fundamental insight provided by the Records Continuum Model is that sound recordkeeping that takes long-term preservation issues into consideration requires proactivity. The case study also shows that the strategy for current recordkeeping i.e. recordkeeping that will mostly be handled by an EDRMS, must be complemented with a strategy that captures, manages and preserves the records in other systems, that have non-document formats.

A natural continuation of this research would be to present design recommendations based upon this case study.

References

- [1] National Archives of Australia, Digital Recordkeeping: Guidelines for Creating, Managing and Preserving Digital Records., N.A.o. Australia, ed, 2004.
- [2] H.M. Gladney, Trustworthy 100-Year Digital Objects: Evidence After Every Witness Is Dead, *ACM Transactions on Information Systems*, **22** (2004), 406-436.
- [3] S. McKemmish, M. Piggott, B. Reed and F. Upward, eds, *Archives: Recordkeeping in Society* Charles Sturt University, Centre for Information Studies Wagga Wagga, 2005, 348 pp.
- [4] T.J. Sprehe, Integrating Records Management into Information Resources Management in U.S. Government Agencies, *Government Information Quarterly*, **17** (2000), 13-26.
- [5] M.B. Nunes, F. Annansingh, B. Eaglestone and R. Wakefield, Knowledge management issues in knowledge-intensive SMEs, *Journal of Documentation*, **62** (2006), 101-119.
- [6] International Standards Organization, *ISO 15489-1. Information and Documentation and Records Management Part 1: General.*, International Standards Organization, Geneva, 2001.
- [7] B. Reed, Records, in: *Archives: Recordkeeping in Society* S. McKemmish, M. Piggott, B. Reed and F. Upward, eds, Charles Sturt University, Centre for Information Studies, Wagga Wagga, 2005, pp. 101-130.
- [8] D. Bearman, Electronic evidence: strategies for managing records in contemporary organizations, *Archives and Museum Informatics*, Pittsburgh, 1994, vi, 314 pp.
- [9] M.L.D. Biagio and B. Ibricu, A balancing act: learning lessons and adapting approaches whilst rolling out an EDRMS, *Record Management Journal*, **18** (2008), 170-179.
- [10] J. Gunnlaugsdottir, As you sow, so you will reap: implementing ERMS *Record Management Journal*, **18** (2008), 21-39.
- [11] H.Z. Henriksen and K.V. Andersen, Electronic records management systems implementation in the Pakistani local government, *Record Management Journal*, **18** (2008), 40-52.
- [12] R. Maguire, Lessons learned from implementing an electronic records management system, *Records Management Journal*, **15** (2005), 150-157.
- [13] D.J. Williams, EDRM implementation at the National Weights and Measures Laboratory *Records Management Journal*, **15** (2005), 158-166.
- [14] K. Gregory, Implementing an electronic records management system: A public sector case study, *Records Management Journal*, **15** (2005), 80-85.
- [15] Z.A. Smyth, Implementing EDRM: has it provided the benefits expected?, *Records Management Journal*, **15** (2005), 141-149.
- [16] L. Wilkins, P.M.C. Swatman and D. Holt, Achieved and tangible benefits: lessons learned from a landmark EDRMS implementation, *Record Management Journal*, **19** (2009), 37-53.
- [17] European Commission, *MoReq2: Model requirements for the management of electronic records*, Office for Official Publications of the European Communities, Luxembourg, 2008, 94 pp.
- [18] Department of Defense, Electronic Records Management Software Applications Design Criteria Standard: DoD 5015.02-STD, U.S.o.A. Department of Defence, ed, Department of Defence, 2007.
- [19] International Council on Archives, Principles and Functional Requirements for Records in Electronic Office Environments - Module 1: Overview and Statement of Principles, International Council on Archives, 2008.
- [20] International Council on Archives, Principles and Functional Requirements for Records in Electronic Office Environments - Module 2: Guidelines and Functional Requirements for Electronic Records Management Systems, International Council on Archives, 2008.

- [21] International Council on Archives, Principles and Functional Requirements for Records in Electronic Office Environments - Module 3: Guidelines and Functional Requirements for Records in Business Systems, International Council on Archives, 2008.
- [22] E.A.M. Borglund, Electronic Records use Changes Through Temporal Rhythms, *Archives & Social Studies: A Journal of Interdisciplinary Research* **2**(2008), 103-134.
- [23] C.M. Dollar, *Authentic electronic records : strategies for long-term access*, Cohasset Associates Inc., Chicago, Ill., 2000, x, 248 pp.
- [24] L. Duranti, The Impact of Technological Change on Archival Theory, in: *International Council on Archives Conference*, September 16, 2000, Seville, Spain., 2000.
- [25] L. Duranti, Concepts, Principles, and Methods for the Management of Electronic Records, *The Information Society*, **17** (2001), 271-279.
- [26] A.J. Gilliland-Swetland and P.B. Eppard, Preserving the Authenticity of Contingent Digital Objects, in: *D-Lib Magazine* Vol. 6, 2000.
- [27] S. McInnes, Electronic Records: the new archival frontier, *Journal of the Society of Archivists*, **19** (1998), 211-220.
- [28] S.-S. Chen, The Paradox of Digital Preservation, *IEEE Computer*, **34** (2001), 24-29.
- [29] R.H. Sprague, Electronic Document Management: Challenges and Opportunities for Information Systems Managers, *MIS Quarterly*, **19** (1995), 29-49.
- [30] G. Walsham, Interpretative Case Studies in IS Research: Nature and Method, in: *Qualitative research in information systems: a reader*, M.D. Myers and D.E. Avison, eds, SAGE, London, 2002, pp. 101-113.
- [31] B. Dahlbom, The new Informatics, *Scandinavian Journal of Information Systems*, **8** (1996), 29-48.
- [32] Trafikverket, Ådalsbanan, 2010, retrieved 26 september 2010 from: <http://www.trafikverket.se/Privat/Vagar-och-jarnvagar/Sveriges-jarnvagsnat/Adalsbanan/>
- [33] T. Thomassen, A First Introduction to Archival Science, *Archival Science*, **1** (2001), 373-385.
- [34] International Council on Archives, ISAD(G) : general international standard archival description : adopted by the Committee on Descriptive Standards, Stockholm, Sweden, 19-22 September 1999, Subdirección General de Archivos Estatales, Madrid, 2000, 120 pp.
- [35] H. Hofman, Lost in cyberspace : where is the record?, in: The concept of record : report from the Second Stockholm Conference on Archival Science and the Concept of Record, 30-31 May 1996, K. Abukhanfusa, ed, Swedish National Archives, Stockholm, 1998, pp. 115-129.
- [36] T.R. Schellenberg, *Modern archives : principles and techniques*, SAA. (Original work published 1956), Chicago, 1956/1998, 247 pp.
- [37] R.J. Cox, *Managing records as evidence and information*, Quorum Books, Westport, Conn.; London, 2001, xv, 243 p. ; 225 cm. pp.
- [38] A. Giddens, *The constitution of society : outline of the theory of structuration*, University of California Press, Berkeley, 1984, 402 pp.
- [39] F. Upward, The records continuum, in: *Archives: Recordkeeping in Society*, S. McKemmish, M. Piggott, R. Barbara and F. Upward, eds, Charles Sturt University, Centre for Information Studies Wagga Wagga, 2005, pp. 197-222.
- [40] F. Upward, Modeling the continuum as paradigm shift in recordkeeping and archiving processes, and beyond - a personal reflection, *Records Management Journal*, **10** (2000), 115-139.
- [41] D.G. Ancona, G.A. Okhuysen and L.A. Perlow, Taking Time To Integrate Temporal Research, *Academy of Management Review*, **26** (2001), 512-529.
- [42] W.J. Orlikowski and J. Yates, It's About Time: Temporal Structuring in Organizations, *Organization Science*, **13** (2002), 684-700.
- [43] M. Reddy and P. Dourish, A finger on the pulse: temporal rhythms and information seeking in medical work in: *Proceedings of the 2002 ACM conference on Computer supported cooperative work* pp.344-353, New Orleans, Louisiana, USA 2002.
- [44] A. Menne-Haritz, Access-the reformulation of an archival paradigm, *Archival Science*, **1** (2001), 57-82.
- [45] International Standards Organization, *ISO 15489-2. Information and documentation — Records management — Part 2: Guidelines*, International Standards Organization, Geneva, 2001.
- [46] F. Upward, The Records Continuum and the Concept of an End Product, *Archives and Manuscripts*, **32** (2004), 40-62.
- [47] E.A.M. Borglund, *Design for recordkeeping: areas of improvement*, Department of Information Technology and Media, Mid Sweden University, Sundsvall, 2008, xiii, 185 s. pp.

Evaluation of the Archetypes Based Development

Gunnar PIHO^{a,b}, Jaak TEPANDI^a and Mart ROOST^a

^a*Department of Informatics, Tallinn University of Technology, Raja St. 15, Tallinn 12617, Estonia*

^b*Clinical and Biomedical Proteomics Group, CRUK, LIMM, St James's Univ. Hospital, Beckett St, Leeds LS9 7TF, UK*

Abstract. Archetypes based development (ABD) utilizes archetypes and archetype patterns to increase dependability of software, reduces semantic heterogeneity of models and data types, improves maturity of the software development process, leads development of one-off software towards software factories, as well as satisfies the needs of small and medium sized software houses. ABD uses Zachman framework (ZF) for enterprise architecture in combination with business archetypes and archetype patterns as a central idea for the engineering of domains, requirements and software. We explain the ABD and evaluate it from the Bjørner's domain modelling, MDA (Model Driven Architecture), XP (Extreme Programming) and CMMI (Capability Maturity Model Integration) for Development perspectives.

Keywords. Archetypes and archetype patterns based development, domain engineering, Zachman Framework (ZF), Model Driven Architecture (MDA), Extreme Programming (XP), Capability Maturity Model Integration (CMMI) for Development.

Introduction

The growing pressure to improve software quality as well as reduce cost and time to market has in recent years catalyzed a transition to more automated software development methods. The Software Factories (SF) [1] approach is one of these automated software development or mass customization methods, which promise greater gains in productivity and predictability by making application assembly more cost-effective through systematic reuse and by enabling the formation of supply chains.

The business archetypes and archetype patterns [2] introduced by Arlow and Neustadt are among the key innovations for developing better software with less resources. Our industrial background in developing software for clinical laboratories has stimulated ongoing research in archetypes based development (ABD) [3; 4; 5; 6]. We investigate the research topics [7] posed by Bjørner, by combining the software engineering triptych [8; 9; 10] methodology with archetypes and the archetype patterns (A&AP) initiative from Arlow and Neustadt [2].

ABD [3; 4] is a software triptych process with business archetypes and business archetype patterns. ABD addresses a number of challenges that software development faces today, such as increasing dependability [11] of developed software, reducing semantic heterogeneity [12] of models and data types, improving the maturity [13] of

the software development process, reducing cost and time to market, and leading to development of one-off software towards Software Factory [1].

In the process of elaborating the principles for ABD it soon became evident that there are a large number of views, issues, and artefacts to be developed and used. It is vital to understand, arrange, and utilize them in a systematic way. The best approach for this is to apply an already established overarching framework for enterprise architecture. There exist various frameworks, and the Zachman Framework (ZF) [14] for enterprise architecture has been widely accepted as a standard for identifying and organizing descriptive representations that have critical roles in enterprise management and system development. For this reason the ZF was selected as a reference model for presenting the various ABD facets.

To enable systematic usage of diverse enterprise architecture views and aspects in ABD, we describe the central idea of ABD (business archetype patterns - party and party relationship, product, inventory, order, quantity and rule [2]), as well as the ABD itself in the context of ZF columns and rows, and present the ZF with business archetypes and business archetype patterns. This ZF with archetypes and archetype patterns helps developers to better understand business requirements, to design cost effective enterprise applications through systematic reuse of archetypal components by enabling supply chains of product families, as well as to justify and explain solutions to business side stakeholders. In this paper we evaluate the presented ZF with archetypes and archetype patterns from the Bjørner's domain modelling, MDA (Model Driven Architecture), XP (Extreme Programming) and CMMI (Capability Maturity Model Integration) for Development perspectives

The paper is organized as follows. In Section 1, we introduce basic concepts and principles. We present the ABD in Section 2, evaluate the ABD in Section 3 and then present our conclusions.

1. Basic Concepts

The main target of ABD is rapid application development (RAD) of product families. While general-purpose RAD uses logical information about the software captured by general-purpose development artefacts, the ABD uses conceptual information captured by domain specific models. ABD focuses on Software Factories, i.e. on family-based development artefacts (domain specific languages, patterns, frameworks, tools, micro processes, and others) that can be used to build the family members. Our ABD process model is based on the software engineering triptych (from domain via requirements to software) model [8; 9; 10] and roughly consists of the following:

- Analysis of the business domains using domain analysis methodology similar to one suggested by Bjørner [10]. We suggest the ZF based approach [3] in combination with archetypes and archetype patterns (Section 2).
- In ABD, as common for Software Factories, all models are software artefacts and not only documentation artefacts. For instance, all our clinical laboratory models – the industrial case study of our research - are realized as .NET class libraries by using test driven development (TDD) [15] methodology.
- We use these domain models as the “ubiquitous language” [16] for prescribing and formalizing requirements from customers. These customer requirements we can validate according to the domain models.

2. Archetypes Based Development and the Zachman Framework

In ABD we use the Zachman Framework (ZF) [14] for enterprise architecture in combination with archetypes and archetype patterns [2] as the main methodology for analyzing and modelling of domains as well as for analyzing and selecting solutions and realizations for customer requirements. The general picture of ZF columns with archetypes and archetype patterns is illustrated in Table 1.

Table 1. ZF columns with archetypes and archetype patterns.

Business requirements						
What	How	Where	Who	When		Why
Things	Processes	Locations	Persons	Events		Strategies
Products and services	Reporting (feedback)	Organization and organization structure	Persons	Business events		Business rules
		Party AP				
Product AP	Party relationship AP			Order AP	Inventory AP	
Rule AP						
Quantity and money AP						
Common infrastructure						

2.1. Each Column is a Generic Model of Archetype Pattern

Each column of the ZF describes “single, independent phenomena” [17], which are Things (What?), Processes (How?), Locations (Where?), People (Who?), Time (When?) and Motivation (Why?). In ABD we are modelling these ZF questions with business archetypes and business archetype patterns as follows.

Column 1 (What, Things) describes how products (either goods or services) are related to each other. Examples of product relations are “produced by using”, “produced from”, “is component of”, “belongs to”, “upgradable to”, “substituted by”, “complemented by”, “compatible with”, “incompatible with”, etc. For modelling of product features and product relationships we use *product archetype pattern* similar to one described in [2]. Two other archetype patterns we need when describing products are *quantity archetype pattern* [2] and *rule archetype pattern* [2].

Column 2 (How, Processes) describes business processes. Naive examples of business processes are “buying”, “selling”, “producing”, also “planning”, “servicing”, “controlling”, “reporting”, “transporting”, “communication”, and so on.

For modelling of business processes we use the reporting or more generally the feedback archetype pattern. By using feedback it is possible to collect information about the processes. Based on *party relationship archetype pattern* [2] we have designed a *progress report archetype pattern* (Figure 1). Each *progress report* (possibly from address to address) is the party relationship where *subordinate* reports (the role of person) to *supervisor* (the role of person) or to some *organization unit role* (accounting department for example).

Column 3 (Where, Location) describes the structure of the organisation in terms of organization units and the roles of each organization unit. Like with people, where we strongly separate roles (patient, client, mother...) from the individual, we also separate roles strongly from organization units. For modelling of locations we use *party* as well as *party relationship* archetype patterns [2].

Column 6 (Why, Strategies) describes the strategies in terms of business rules. We use the simple propositional calculus based *rules archetype pattern* [2] as the basic model for strategies.

2.2. Each Row is a Perspective of Triptych Development

The triptych software process – from domain model via requirements to software – has a very simple informal description: before writing software, we have to know requirements; before we know requirements, we have to understand domain; to understand the domain we have to study one.

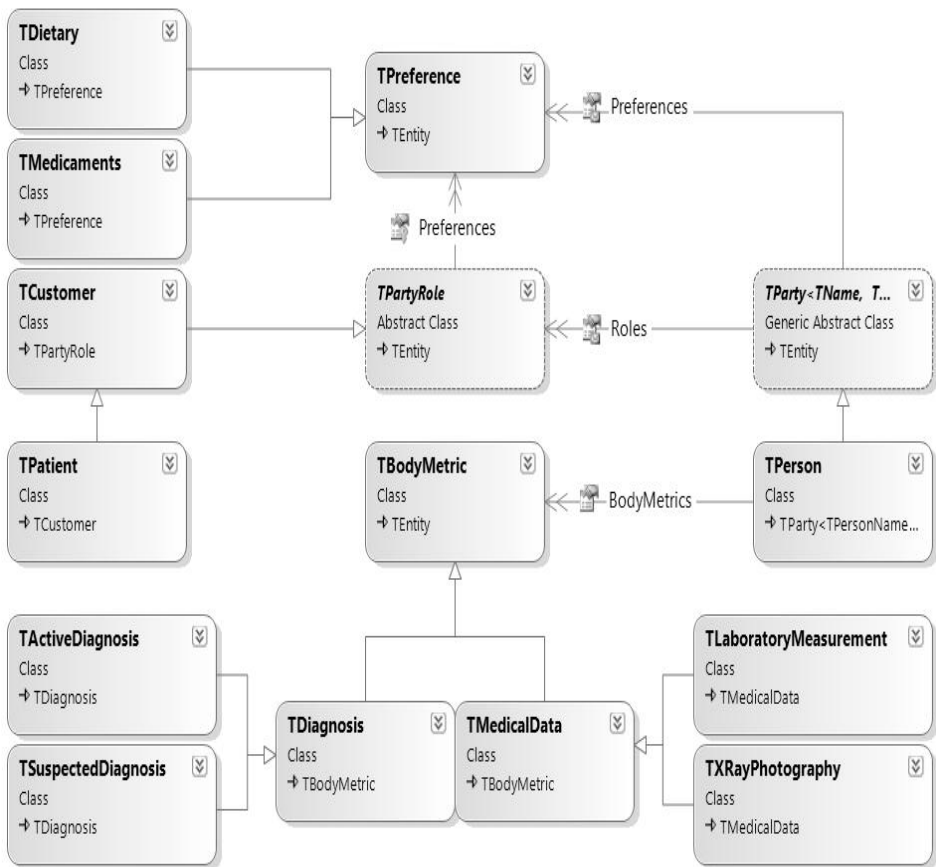


Figure 2. Patient Archetype.

The interpretation on ZF rows in terms of triptych (requirements, domain, and software) development can be as follows:

Row 1 (Conceptual model) is just the glossary (list of things, objects, assets, etc) that defines the scope or boundary of enterprise requirements. For example, the cell defining the scope for column 4 (people) for the clinical laboratory can include terms like *patient*, *clinician*, *medical technical assistant*, and so on.

Row 2 (Semantic model) is a definition and a model of the actual enterprise requirements. It defines the concepts (terms and facts) that the enterprise actually needs. This can be represented as simple narratives (for instance “patient has a hospital number”, “patient has a clinician”, etc) or in some more formalized (class diagrams for example) notation.

Row 3 (Logical model) describes the requirements in terms of the domain model. For column 4 this means for example, that “patient is the role for person”, “patient hospital number is a registered identifier for a person whose current role is patient”, etc. Figure 2 illustrates how patient and patient related information, like diagnosis (actual and suspected), medical data (laboratory determinations and etc), dietary restrictions and medicaments can be modelled by using archetypes from *party archetype pattern* [2]. *TParty*, *TPerson*, *TPartyRole*, *TBodyMetric* and *TPreferences* are archetypes defined by party archetype pattern. According to *party archetype pattern* each person (*TPerson*) has zero or more body metrics (*TBodyMetric*) and each party (*TParty*) as well as each party role (*TPartyRole*) has zero or more preferences (*TPreference*). Laboratory is a place equipped for performing tests or doing experimental work. A clinical laboratory (or medical laboratory) is a laboratory where tests are performed on clinical specimens for the purpose of providing information on diagnosis, prognosis, prevention, or treatment of disease. In Figure 2 the patient (*TPatient*) is derived from party role (*TPartyRole*) archetype. All kinds of measurements (*TMedicalData*) (laboratory measurement, X-ray photography’s and etc) as well as all kinds of estimates, diagnosis (*TDiagnosis*) for example, we have modelled as body metrics (*TBodyMetric*) of a person. All recommendations (dietary, treating, etc) we have modelled as preferences (*TPreference*)

Row 4 (Physical model) is the actual model of the domain. For column 1 (what) this is the definition of *product archetype pattern*; some *party relationship archetype pattern* derivate like progress report for business processes (column 2, how); party and party relationship archetype patterns for enterprise structure (column 3, location, where); party and party relationship archetype patterns for stakeholders (column 4, who); order and inventory archetype patterns for business events (column 5, when); and rule archetype pattern for business rules (column 6, why).

Row 5 (Detailed definition) is the realization of archetype patterns (product, party, party relationship and derivatives, order, inventory, quantity and rule) as APIs and as database schemas supporting this pattern under some concrete database engine.

Row 6 (Product) is the software or service which uses the domain model (described according to row 4 and implemented according to row 5) and fulfils the requirements scoped by row 1, explained by row 2 and specified in terms of archetypes and archetype patterns (row 3).

2.3. Testing the Approach with the LIMS Software Factory

We are validating our approach through using ABD in the development of a Laboratory Information Management System (LIMS) [18] Software Factory [1]. Development of the various LIMS Software Factory artefacts was made more systematic and explicit by using the arrangement provided by the Zachman Framework with archetypes and archetype patterns.

Based on the domain model (describes and abstracts the universe of discourse as it is) of the clinical laboratory (implemented as API) the LIMS Software Factory architecture (Figure 3) consists of LIMS DSL (domain specific language), LIMS

Engine, and Tests Engine. With the LIMS DSL it should be possible to describe and modify the requirements for particular LIMS software. According to these requirements the LIMS Engine must generate the LIMS software and according to test scenarios (acceptance tests, generated also by LIMS DSL), the Tests Engine should validate the requirements according to the domain model and verify that the LIMS software meets requirements.

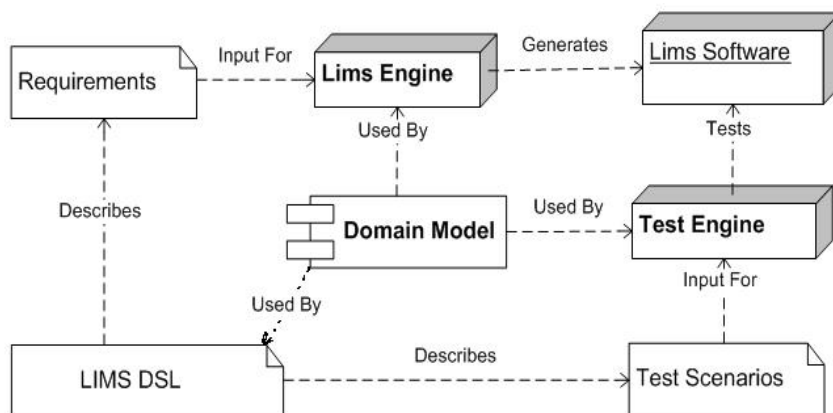


Figure 3. Architecture of LIMS software factory.

In the current stage of project, we have focused on the domain model and on the possibility of describing the requirements of the laboratory (service layer) by using terminology and features from this domain model. We are also investigating the possibility of automating the generation of the UI as well as database schemas based on the domain model. The working prototype is currently being used by three different research groups inside the Clinical and Biomedical Proteomics Group, Cancer Research UK Clinical Centre, Leeds Institute of Molecular Medicine. Each of the groups has their own particular business requirements and needs. The target is not to implement the LIMS, which satisfies the needs of all research groups, but to implement the LIMS so that each research group (maybe in the future even at run-time) can customize the LIMS (different features, rules, data structures and layouts, business procedures and research methods, etc) according to their needs.

3. Evaluation of ABD

In the following section we evaluate the ABD by comparing it with Bjørner's domain modelling methodology, Model Driven Architecture (MDA), Extreme Programming, as well with Capability Maturity Model of Integration (CMMI) for Development.

3.1. ABD and Bjørner's Domain Modelling

Bjørner's domain analysis and modelling methodology [10] describes how to develop the domain model and is based on domain stakeholders as well as on pragmatically chosen domain facets. According to Bjørner, the domain facets are intrinsic, business

processes, supporting technologies, management and organization, rules and regulations, scripts and human behaviour. Our ABD is based on ZF (Zachman framework) for enterprise applications and on respective questions: Who, What, When, Where, Why, and How.

Table 2. ABD and Bjørner’s Domain Analysis Methodology

	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
Zachman Framework	What (Things)	How (Processes)	Where (Location)	Who (Persons)	When (Events)	Why (Strategies)
Archetypes Based Development	Product AP	Progress Report AP	Party and Party Relationship AP		Order and Inventory AP	Rule AP
Bjørner’s Domain Analysis	Intrinsics	Main Business Process; Related Processes	Management and Organization	Stakeholders; Human Behaviour		Rules, Regulations and Scripts

We argue that the Bjørner’s domain facets based domain analysis method is a special case of domain analysis methodology based on Zachman framework with archetype patterns (Table 2). For example the set of intrinsic concepts (basic concepts to any other domain facet) is the subset of all products and services the businesses use, buy or sell that can be analyzed and modelled by using product archetype pattern (column 1). The main business process together with related processes can be analyzed and modelled by reporting (or feedback) archetype pattern (column 2). The management and organization can be analyzed and modelled with party archetype pattern and party relationship archetype pattern (column 3). The stakeholders as well as human behaviour are part of party and party relationship archetype patterns describing persons (who) in businesses (column 4). The rules, regulations and scripts can be analyzed and modelled according to the rule archetype pattern (column 6). In addition to the Bjørner’s facets, the ZF based approach has order and inventory archetype patterns for analyzing and modelling events (column 5).

3.2. ABD and MDA

Table 3 summarizes how in our opinion the software triptych, ZF and Model Driven Architecture activities (MDA) [19] are related.

Table 3. The rows of ZF in the context of software triptych and MDA

Triptych	ZF Rows	MDA
Requirements	1,2	CIM (conceptual, business, analysis)
Domain	3,4	PIM (logical, design)
Software	5,6	PSM (physical, implementation)

The upper (first and second) rows of ZF correspond to the requirements from some concrete enterprise in the context of the software triptych and to the Computing Independent Model (CIM) in the context of the MDA. We interpret the CIM also as the conceptual and business level model that is a product of the enterprise requirements analysis process. The middle (third and fourth) rows of ZF correspond to the domain part in the context of the software triptych and to the Platform Independent Model

(PIM) in the context of MDA. The PIM is interpreted as a logical design model. The lower (fifth and sixth) rows of ZF correspond to software part in the context of the software triptych and to the Platform Specific Model (PSM) in the context of MDA. The PSM is a physical design and implementation both of domain (row 5 in ZF) and of requirements (row 6 in ZF) described in domain specific language (row 3 of ZF) which is based on logical design (row 4 of ZF) of archetypes based domain model.

3.3. ABD and XP

Extreme Programming (XP) [20] is an agile software development methodology with basic practices like test driven development, pair programming, planning game, continuous integration, small releases, metaphor, simple design, refactoring, collective ownership, 40-hour week, coding standards and so on. Table 4 summarizes ABD and XP activities.

Table 4. Comparing XP and ABD (made by using similar table from [21])

Commonsense	XP extreme	XP practice	ABD for SF	ABD for Software
Manage requirements	Review requirements all the time	On site customer	On site domain specialist	Requirements are coded in DSL so that customer can validate them
Code reviews	Review code all the time	Pair programming	Pair programming from XP can be used	Code is automatically generated (mostly)
Testing	Test all the time, even by customers	Unit testing, functional testing	Unit tests based domain modelling	Domain model validates and verifies requirements (automatically)
Design	Make design part of everybody's daily business	Refactoring	Refactoring towards archetype (general and domain specific) patterns	Archetypes and archetype patterns based predefined by SF design
Simplicity	Simplest design that supports the system's current functionality	The simplest thing that could possibly work	The simplest abstraction that could possibly work	
Architecture	Everybody works to refine the architecture all the time	Metaphor	Based on ZF with archetype patterns	Archetypes and archetype patterns based predefined by SF architecture
Integration testing	Integrate and test several times a day	Continuous integration	Continuous integration from XP can be used	
Short iterations	Short (seconds, minutes, and hours) iterations	Planning game	Archetypes based planning (game)	
Manage versions	Plan and release frequently small units of business functionality	Frequent small releases	Archetype patterns based releases	Requirements based step by step releases with possibilities to undo and redo.

While XP is for developing tailored one-off software for customer and is based on customer requirements, the ABD is for developing software factories (SF) so that tailored one-off software for specific customer requirements can be generated automatically by using SF tools and other artefacts. In view of this, Table 4 (composed

using a similar table from [21]) has two columns for ABD. The "ABD for SF" column summarizes how to use XP activities when developing software factory artefacts. The "ABD for Software" column summarizes the activities needed for generating software from SF according to customer needs.

When developing archetypes and archetype patterns based software factory artefacts, we mostly use XP practices in combination with domain analysis and domain modelling activities. So instead of an "on site customer" we need a domain specialist. Instead of XP practices, where everyone can change design (refactoring) as well as refine the architecture (metaphor) towards simplest thing and design that possibly works, in ABD, when developing SF artefacts, we have mostly fixed ZF with archetype patterns based architecture. Therefore the refactoring is mostly towards efficient and universal archetype patterns; this is where also the design [22] and application architecture [23] patterns come into account.

ABD uses the XP unit testing practice in domain analysis and modelling. This means, that all domain narratives are specified as unit tests [15]. We call this approach as unit test based domain modelling.

When the SF is ready, we can generate one-off software automatically (Figure 3). This means that by using a domain model based DSL (domain specific language) we "code" the customer requirements. The DSL must be designed so that a domain specialist is able to understand this DSL and is able to validate the resulting requirements. The software generated will be based on these requirements. The requirements will be first validated according to the domain model and the generated software will be verified according to requirements as well as according to domain model. As the final validation and verification can be conducted only when the software is deployed into the real environment and used by the customer in real everyday business, it is wise to implement and deploy requirements step-by-step. For these purposes the undo and redo mechanisms for requirements as well as for data must be implemented in SF artefacts.

The ABD complements the XP by focussing on the understanding of the domain (what to do) and on modelling domains formally, on the decision analysis and resolution (by selecting a solution that meets the multiple demands of relevant stakeholders), on the requirements development (by describing the customer requirements in terms of domain), and on validation and verification (by validating and verifying requirements against domain model).

3.4. ABD and CMMI for Development

CMMI (Capability Maturity Model Integration) for Development "*is a process improvement maturity model for the development of products and services*" [13]. It describes best practises for improving maturity of software development. In what follows, we refer to CMMI for Development as to "CMMI".

ABD addresses many of the CMMI Level 2 *requirements management* process area (PA) specific practices through its use of the domain model, synopsis (similar to stories in XP) and narratives (similar to XP tasks). When XP integrates feedback on customer expectations and needs by emphasizing short release cycles and continual customer involvement, the ABD maintains "common understanding" through the ZF with archetypes and archetype patterns by asking questions what, how, where, who, when and why. Although the requirements from the customer might evolve dramatically over time, in our understanding the properly abstracted and formalized

domain model simplifies the introduction of changes to the specifications, as requirements are in terms of domain model. In addition, it reduces the risks involved with introducing these changes, as test driven modelling enables (at least partially) to verify and validate user requirements according to domain model.

In ABD the project plan (*project planning PA*) is not detailed for the project whole life cycle, although the archetype patterns based system architecture establishes the project's main direction. As a result, by analyzing and designing requirements in terms of archetype patterns based domain model together with short iterations (1-3 weeks) and small releases (2-6 months) enables the developers to identify and manage plans efficiently. The customer maintains control of business priorities by choosing which requirements to implement within given resources.

The *project monitoring and control PA* (to provide adequate visibility into actual progress) and the *measurement and analysis PA* (to provide practices that guide projects and organizations with objective measurements of processes, work products and services) are related. ABD addresses project monitoring similarly to XP by using "big visual chart", project velocity, and commitments for small releases [21]. The "big visual chart" in XP means an open workspace together with white board based information reflecting the projects progress and close communication between project members also onsite customer. An overall schedule and budget in XP are calculated by figuring the estimated time for the work factored with the project velocity (40-hours weeks, implemented tasks per developer per week and etc.). By using of small releases, the feedbacks for commitments from real users from the real environment provide reassurance and the opportunity to intervene fast. All these activities are also ABD activities. Differently from XP, where the development team is mostly like an explorer with a true compass, the ABD team is also equipped with a decent map – the archetypes and archetype patterns based domain model – which gives additional possibilities (where we are, how much is to go) also for measurement and analysis both work products as well as development processes.

ABD addresses Level 3's *risk management PA* (manage risks with continuing and forward-looking activities that include identification of risk parameters) partly through its activities described already in project monitoring and control PA. Additionally some preventative activities like customer readable simple synopsis and narratives, archetypes and archetype patterns based design, refactoring, coding standards, unit testing and especially unit testing based modelling are all elements of risk management.

Requirements development PA (identifies customer needs and translates these needs into high-level conceptual solutions) is addressed in ABD through describing customer requirements in terms of the archetypes based domain model. Translation of customer requirements into domain model terms is one of the key features of ABD. The archetypes based domain model is also the key feature that addresses Level 3's *product integration* (generate the best possible integration sequence by integrating product components) and the *technical solution* (develops technical data packages for product components) PA-s. The same is also true for *validation* (incrementally validate products against the customer's needs) as well for *verification* (ensure that selected work products meet the specified requirements) which are both the natural components of ABD. By formal analysis of requirements through using of archetypes and archetype patterns based domain analysis and modelling techniques the subjective nature of requirements, design and architecture decisions will be reduced in order to select solutions that meets multiple demands of relevant stakeholders.

4. Conclusions

While implementation and testing of the LIMS Software Factory proves feasibility of ABD in real life systems, the above analyses also demonstrates that ABD is in agreement with and complements important software development processes and methodologies, such as Bjørner’s domain modelling, MDA (Model Driven Architecture), XP (Extreme Programming) and CMMI (Capability Maturity Model Integration) for Development.

Table 5. ABD satisfaction of CMMI process areas

Level	Key process areas	Satisfaction
2: Managed	Requirements Management	++
	Project Planning	+
	Project Monitoring and Control	+
	Measurement and Analysis	+
	Process and Product Quality Assurance	+
	Configuration Management	-
	Supplier Agreement Management	-
3: Defined	Organizational Process Focus	-
	Organizational Process Definition	-
	Organizational Training	-
	Integrated Project Management	-
	Risk Management	+
	Decision Analysis and Resolution	-
	Requirements Development	++
	Product Integration	++
	Technical Solution	++
	Validation	++
	Verification	++
4: Quantitatively Managed	Organizational Process Performance	-
	Quantitative Project Management	-
5: Optimized	Organizational Innovation and Deployment	-
	Causal Analysis and Resolution	-

++ largely, + partly and - not addressed in ABD

ABD complements XP by focussing on understanding the domain (what to do) and on modelling domains formally, on the domain analysis (by selecting a solution that meets the multiple demands of relevant stakeholders), on the requirements development (by describing the customer requirements in terms of domain), and on validation and verification (by validating and verifying requirements against domain model). As shown in Table 5, by using ABD it is possible to cover some institutional practices that the CMMI for Development identifies as key elements for good engineering and management.

Acknowledgments

This work is supported by Estonian Ministry of Education and research (SF0140013s10); by Estonian Science Foundation (grant 6839); by Tallinn University

of Technology (Estonia); by University of Leeds (United Kingdom); by Cancer Research UK.

References

- [1] Greenfield, J., et al., *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. s.l. : Wiley, 2004.
- [2] Arlow, J. and Neustadt, I., *Enterprise Patterns and MDA: Building Better Software With Archetype Patterns and UML*. s.l. : Addison-Wesley, 2003.
- [3] Piho, G., Tepandi, J. and Roost, M., "The Zachman Framework with Archetypes and Archetype Patterns" [ed.] J. Barzdins and M. Kirikova. Riga, Latvia, Baltic DB&IS, July 5-7, 2010 : University of Latvia Press, 2010. Databases and Information Systems: Proceedings of the Ninth International Baltic Conference. pp. 455-570.
- [4] Piho, G., Roost, M., Perkins, D., and Tepandi, J., "Towards archetypes-based software development" [ed.] T. Sobh and K. Elleithy. s.l. : Springer, 2010. Innovations in Computing Sciences and Software Engineering: Proceedings of the CISSE 2009 . pp. 561-566. DOI: 10.1007/978-90-481-9112-3_97 . ISBN: 978-90-481-9111-6.
- [5] Piho, G., Tepandi, J., Perkins, D., Parman, M., "From archetypes-based domain model of clinical laboratory to LIMS software" Opatia, Croatia, 24-28 May 2010 : IEEE, 2010. MIPRO, 2010 Proceedings of the 33rd International Convention. Vol. Digital Economy, pp. 1179-1184. ISBN: 978-1-4244-7763-0.
- [6] Tepandi, J., Piho, G. and Liiv, I., "Domain Engineering for Cyber Defence Visual Analytics: a Case Study and Implications" Tallinn, Estonia : CCD COE Publications, 2010. CCDCOE Conference on Cyber Conflict. pp. 59-77.
- [7] Björner, D., "Domain Theory: Practice and Theories (A Discussion of Possible Research Topics)" Macau SAR, China : The 4th International Colloquium on Theoretical Aspects of Computing - ICTAC, 2007.
- [8] Björner, D., *Software Engineering, Vol. 1: Abstraction and Modelling*. Texts in Theoretical Computer Science, the EATCS Series. : Springer, 2006.
- [9] Björner, D., *Software Engineering, Vol. 2: Specifications of Systems and Languages*. Texts in Theoretical Computer Science, the EATCS Series. : Springer, 2006.
- [10] Björner, D., *Software Engineering, Vol. 3: Domains, Requirements, and Software Design*. Texts in Theoretical Computer Science, the EATCS Series. : Springer, 2006.
- [11] Avižienis, A., Laprie, J.-C. and Randell, B., *Fundamental Concepts of Dependability*. Research Report N01145. s.l. : LAAS-CNRS, April 2001.
- [12] Halevy, A.Y., "Why Your Data Won't Mix: Semantic Heterogeneity." *Queue*. 2005, Vol. 3, 8, pp. 50-58.
- [13] CMMI for Development, Version 1.2, CMU/SEI-2006-TR-008. Software Engineering Institute, 2007. <http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html>.
- [14] Zachman, J. A., "A Framework for Information Systems Architecture." *IBM Systems Journal*. 1987, Vol. 26, 3.
- [15] Beck, K., *Test-Driven Development: By Example*. Boston, MA : Addison-Wesley, 2003.
- [16] Evans, E., *Domain-Driven Design: Talking Complexity in the Heart of Software*. Boston, MA : Addison-Wesley, 2004.
- [17] Zachman, J. A., *The Framework for Enterprise Architecture – Cell Definitions*. s.l. : ZIFA , 2003.
- [18] ASTM., E1578-06 *Standard Guide for Laboratory Information Management Systems (LIMS)*. s.l. : ASTM International, 2006.
- [19] Alhir, S. S., "Understanding the Model Driven Architecture (MDA)." [Online] 2003. <http://www.methodsandtools.com/archive/archive.php?id=5>.
- [20] Beck, K., *Extreme Programming Explained: Embrace Change*. s.l. : Addison-Wesley, 2000.
- [21] Paulk, M. C., "Extreme Programming from a CMM Perspective." Raleigh, NC 23-25 July : Paper for XP Universe, 2001. <ftp://ftp.sei.cmu.edu/pub/documents/articles/pdf/xp-from-a-cmm-perspective.pdf>.
- [22] Gamma, E., et al., *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA : Addison-Wesley, 1995.
- [23] Fowler, M., *Patterns of Enterprise Application Architecture*. Boston, MA : Addison-Wesley, 2003.

Comparison of Plan-driven and Agile Project Management Approaches: Theoretical Bases for a Case Study in Estonian Software Industry

Marion LEPMETS^a and Margus NAEL^b

^a*Institute of Cybernetics, Tallinn University of Technology*

^b*Department of Computer Engineering, Tallinn University of Technology*

Abstract. There is evidence that Scrum could benefit from additional plan-driven practices of defined processes that address the organizational context. The question remains however how to add the plan-driven practices without losing the agility of Scrum and increasing the project performance. This paper describes the preparatory phase of the research that aims to evaluate the tailored project management process in industry. First, we compare the plan-driven project management practices derived from the process models of CMMI and ISO/IEC 15504, the PMBoK, project management literature with the Scrum practices from the case company. We then evaluate the comparison addressing the concerns of using Scrum in industry focusing on the value the plan-driven practices would have in agile environment. Finally, we propose a tailored project management process and suggest measures for evaluating the process in an industry case study.

Keywords: Scrum, CMMI, ISO/IEC 15504, PMBoK, project management, plan-driven methods.

Introduction

Agile software development is a way of organizing the development process, emphasizing direct and frequent communication – preferably face-to-face, frequent deliveries of working software increments, short iterations, active customer engagement throughout the whole development life-cycle and change responsiveness rather than change avoidance. Agile software development can be seen as a philosophy and several defined methods based on these ideas are in use, all sharing a common set of values and principles. The best known and most used agile methods are Extreme Programming (XP) and Scrum [1]. The goal of agile software development is to increase the ability to react and respond to changing business, customer and technological needs at all organizational levels. Agile software development methods are used in a hope of nearing toward this goal [2]. Agile methods have been applied in industry for many years now without them being given much attention in research [3]. In our paper we are trying to fill a gap by focusing on how agile practices relate to established fields in software engineering. We do that by relating Scrum project management practices of a software development company with the already established practices of project management from process models.

Scrum has attracted significant attention among software practitioners during last five years. Whereas the Extreme Programming method that has been widely accepted as one of the most important agile approaches has a definite programming flavour, Scrum concentrates on managing software projects [4]. Scrum was first described by Ken Schwaber and starts with the premise that software development is too complex and unpredictable to be planned exactly in advance [5]. Scrum is an empirical approach based on flexibility, adaptability and productivity. It leaves open for the developers to choose the specific software development techniques, methods, and practices for the implementation process. It involves frequent management activities aiming at consistently identifying any deficiencies or impediments in the development process as well as in the practices that are used [2]. The environmental and technical variables like time frame, quality, requirements, resources and tools must be controlled constantly in order to be able to adapt to changes flexibly. This is achieved through an iterative and incremental development process.

It is argued that focusing on people will improve software productivity and quality [3]. That is exactly what agile does and why it has become so popular. At the same time, agile methods generally lack practices and guidance for implementing and supporting an agile approach across the organization. It is argued that an agile implementation will not “stick” without an organizational context that supports process definitions that are described in Capability Maturity Model Integration [6]. As Boehm and Turner put it in [7] agility without discipline is the unencumbered enthusiasm of a startup company before it has to turn a profit. The discipline of plan-driven methods approach development with standard, well-defined processes that organizations improve continuously.

Software process improvement (SPI) is an applied academic field rooted in software engineering and information systems disciplines, which has been studied for almost twenty years now. It deals primarily with the professional management of software firms, and the improvement of their practice, displaying the managerial focus rather than dealing with the techniques that are used to write software [8]. SPI is based on process assessment where practices of a software company are evaluated against the requirements and practices of process models. There are currently dozens of process models for assessing and improving software development and its related practices. CMMI (Capability Maturity Model Integration) for Development, ISO 9001, ISO/IEC 15504 (IS 15504), ISO/IEC 12207, ISO/IEC 15288 are only a few of the popular ones. In this study we focused on IS 15504 and CMMI as they are widely used in software industry for process assessment purposes. IS 15504 is the only international standard for process assessment at the moment. CMMI, whose underlying ideas have endured over time, has evolved from the concepts of software maturity framework that was developed by Software Engineering Institute already in 1986 and is used extensively today.

As was stated in [4] Scrum could be tailored to be more compliant with CMMI and CMMI model can be improved by adding some Scrum practices on their activities. The question remains how to tailor it so that the best practices of the process models are added to Scrum without losing the agility in its software project management.

The aim of this research is to find out whether combining the practices of plan-driven and agile methods in project management will increase project performance. In order to attain the aim of the study, a comparison of project management practices was carried out based on process models of CMMI for Development v.1.2 [9], IS 15504 [10], Project Management Body of Knowledge (PMBok) of PMI [11] and the project

management practices from Scrum environment. The measures of project performance are also suggested which will be used for evaluating the tailored Scrum process in industry. This paper illustrates the first phase of the research in which the bases for tailoring Scrum project management process is provided.

1. Background and Motivation

Scrum is a product-centered and teamwork oriented way of working that does not describe rules or activities about project initiation or finalizing. Most of the tasks in a software development project are assigned and completed within the team. Glazer et al. claim in [6] that agile methods do not have an organizational context that is described in CMMI. Organizational context supports a long-term view of process adaptation across the enterprise as a whole. For example, knowledge sharing on organizational level through project postmortem reviews would increase the project performance of future projects. The goal of the postmortem reviews is to transfer the experiences of project teams into immediate and concrete software process improvements [12]. The retrospective meetings of Scrum are carried out within the team after every iteration, but no retrospective meeting is held after the development project is finished. The postmortem review at the end of the entire development project allows dissemination of project's experiences to wider audience of developers and managers in the organization. As long as the causes of failed projects are not analyzed, understood and communicated, it is unlikely that the project performance can improve in subsequent projects [13].

Plan-driven methods are characterized by heavy upfront planning, focus on predictability and documentation. Scrum, on the other hand, relies on tacit knowledge within a team as opposed to documentation [7]. There are no budget and status reports required in Scrum environment for progress reviews, for example. Although the software department of a company might share the cultural values similar to agile principles, which is necessary for successful adoption of agile methods [14] and the management approve the adoption of agile methods, the members of the board and management might still want to bring themselves up-to-date with the development projects of the company through various progress reports. For mutual understanding about the project progress, there is a need for standardized indication for the effort, time and budget consumed in the project at any certain time. This kind of deliverables like project status reports have been described in detail as a part of project management process in the plan-driven methods.

These examples are valid in our case company and are supported by relevant related literature motivating this study in tailoring Scrum activities with defined processes of project management.

2. Related Research

There is currently a lot of research conducted on software process improvement and plan-driven software development but it is argued in [3] that agile methods have been given little attention in research despite them having gained industrial acceptance. They claim that management oriented approaches such as Scrum is an example of an area where there is a large gap between industrial acceptance and coverage in research.

They also suggest research to understand how various practices and recommendations in agile development relate to established fields on various issues like project management.

The current research addresses both the topic of agile software development method Scrum and the established field of project management. The latter is viewed based on literature review, the PMBoK and process models of software process improvement area, described in greater detail in [15]. The detailed practices of Scrum are described based on Scrum rules of Scrum alliance [16] that are followed in software development projects of the case company. We describe below the four research articles that are closely related to this study, focusing on agile or, more specifically, Scrum, and the process models.

Tor and Hanssen describe the potential benefits of combining ISO9001 and agile methods in [1]. They conclude that the main difference between ISO 9001 and agile methods is that ISO 9001 insists on documentation for reviews and to demonstrate process conformity. Agile methods try to avoid writing documents that does not contribute to the finished system. On the other hand – if the customer requires a certain document, the use of agile methods are no hindrance for developing them.

Glazer et al. have written a report [6] about embracing both CMMI and agile where they claim that combining the benefits of agile and CMMI will dramatically improve business performance. The report describes in detail CMMI and the agile methods, providing a thorough paradigm comparison and concludes that practitioners should make the best use they can of both of these paradigms to encounter the betterment of their project and organization. This report is a general description of the paradigms and is not aiming to describe any agile method in particular. After having described both paradigms, it suggests that there should be benefits in using a combination of them.

Marçal et al. have mapped the project management practices of CMMI and Scrum in [4]. Their paper shows how Scrum addresses the process areas of project management of CMMI.

Turner and Jain described the results of a workshop activity in [17] where the characteristic of two approaches, agile and CMMI were compared, resulting in a broad mapping between the two approaches. The comparison was made by a group whose members had expertise in both agile and CMMI and remains on the same level of abstraction as the paper by Marçal et al.

These mappings provide valuable insight into the differences and similarities of the two paradigms. Our study differs from the previous ones as we take the comparison from the conceptual to the practice level.

3. Research Method

This study is characterized as analytical and evaluative research and it mainly follows the constructive research approach. Based on [18] there are two processes in the constructive approach - first, the process of building a construct, artifact or model; and second, the process of determining how well the construct, artifact or model performs. In this study we compare the project management practices, tailor the Scrum process and set the measures for evaluating the tailored process in industry case studies.

In this research the motivation comes from industry where additional plan-driven practices need to be combined with their Scrum practices in order to improve their estimation accuracy and project performance in software development.

The case company in our study is a medium-sized [19] software development company that has applied Scrum systematically from 2008 and in four development projects since. The average size for the team has been five people and the average size of project has been six months. A survey among the software developers and project managers in the company has resulted in a set of concerns while using Scrum and has motivated the study of comparing the Scrum practices with the plan-driven practices. The questions were open-ended and targeted to the impediments or shortages of Scrum. The survey results indicate that the information was kept in Scrum teams instead of sharing the knowledge across different organizational levels. This led to the developers missing important technical knowledge and the managers not knowing all the project details. The survey findings suggest that the biggest impediment of the team-centered Scrum process is that it does not support information spreading outside the project.

4. Comparison of Scrum and Plan-driven Project Management Practices

In this chapter we describe the basis of comparison between the project management practices. First, the Scrum practices have been described according to Scrum rules of Agile alliance that are implemented in the software development projects of our case company. We then illustrate the set of plan-driven project management practices that have been derived from CMMI, IS 15504, the PMBoK and project management literature and grouped together in [15].

4.1. Scrum Practices

In this chapter we describe Scrum through its rules, roles and artifacts. We also describe the project management practices of Scrum that are based on the case company using Scrum in their development projects.

Scrum is a product-centered people-oriented framework for project management. It applies iterative and incremental approach to optimize the risks and predictability. The Scrum process is transparent and its frequent deliveries and inspections ensure the highest predictability. The Scrum framework includes roles, rules, artifacts and time-boxes. Time-box is a period of time in which to accomplish a task. Every role and activity in Scrum is time-boxed and thus it improves the velocity of the work. Every role and activity has a set of rules that together will support the achievement of the expected results of the project. Thus, the rules connect the roles, artifacts and time-boxes.

There are three roles in Scrum: the ScrumMaster, the Product Owner and the team. ScrumMaster is responsible for managing the Scrum process. He coaches and leads the team to become better in Scrum, but nonetheless the team is self-organizing and ScrumMaster is not the head of the team like a project manager in plan-driven development project.

Product Owner is the only person responsible for managing the product feature list. He drives the product vision and is the only person who tells the team what to do. A typical Product Owner is either a service, product or business manager.

The team is cross-functional set of developers who turn the product feature list into potentially shippable set of functionalities. The team is self-organizing, i.e. it finds the best way all by itself how to turn the list of features into a shippable product while facing the impediments and challenges as one. The best team size of Scrum is from five to nine persons.

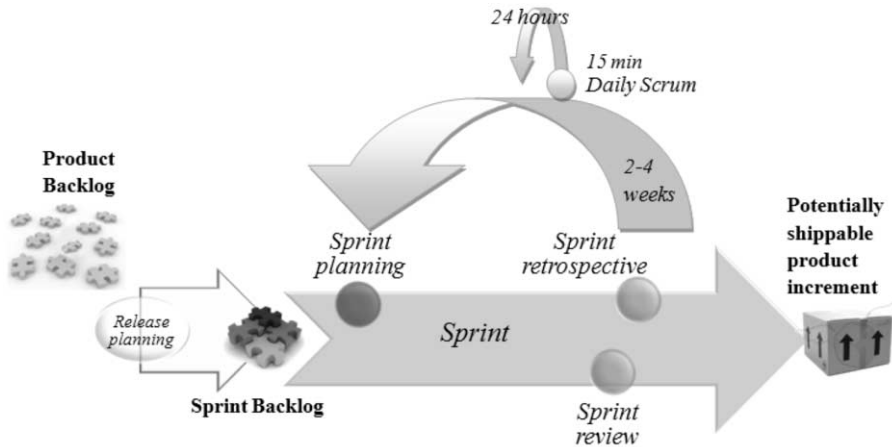


Figure1. The Scrum process (activities in *Italic*, artifacts in **Bold**)

The Scrum process illustrated on Figure 1 consists of five time-boxed meetings and the work period itself. The process is iterative and the iterations are called sprints. The product is developed iteratively, wherein each sprint delivers a set of highest priority and riskiest features or functionality of the product. The product is released when enough sprints have been done, i.e. when enough value has been added sprint by sprint to the product.

In the beginning of the project or release there is a release planning meeting where the team, the Product Owner and ScrumMaster participate. The purpose of the meeting is to set up a release plan and it does not take more time than 15-20% of the time that an organization typically consumes to build a release plan.

The iterative part of the project is the sprint, which contains the following activities: the sprint planning meeting, the development work, the sprint review and the sprint retrospective. The sprint planning meeting is held to set the sprint goal, i.e. what and how the development work will be carried out. The sprint goal is the subset of the release goal and is set according to the data of various inputs like the product backlog, the latest increment of the product, the capability and the past performance of the team.

The product backlog is a list of all features, technologies, functions and improvements that will be made to the product. Product backlog is evolving constantly through the time. Every item in the product backlog is described through three attributes: the description of the item, priority of the item and an estimate for the item being developed.

Only the team decides what and how the work is done in the sprint. Sprint review is an informal meeting where the team and stakeholders describe what was done and what should be done next. The sprint retrospective meeting is for process improvements as the team decides what should be done better in the next sprint. During the development process there are daily stand-up meetings of 15 minutes called the daily Scrums. The purpose of the daily Scrums is to improve communication and

transparency. Every team member tells what he accomplished since the last meeting, what he will do before the next meeting and what the impediments on the way are.

In addition to roles and time-boxed activities, there are four artifacts: the product backlog (PB), release burndown chart (RBD), the spring backlog (SB) and the sprint burndown chart (SBD). Product backlog lists the features of the product. Release burndown chart illustrates graphically the sum of remaining effort according to the estimates in the product backlog. The Product Owner is responsible for updating the graph throughout the project. Sprint backlog lists all the tasks that the team performs to reach the sprint goal and the sprint burndown chart describes the effort remaining for the goal to be reached.

According to the company in our case study, the Scrum practices can be listed based on the tasks of different Scrum roles and meetings. The numbers have been added for each listed practice to make the grouping of practices easier for the comparison.

1. ScrumMaster (activities of)
 - 1.1. Monitor sprint work (through updating sprint burndown chart)
 - 1.2. Schedule control (through updating sprint burndown chart)
 - 1.3. Remove organizational impediments that impede the Scrum
2. Product Owner
 - 2.1. Create and maintain the product backlog (list of features and tasks during product development)
 - 2.2. Set priorities to every item in the product backlog
 - 2.3. Set the acceptance criteria to every item of the product backlog
 - 2.4. Do the acceptance testing to the shipped product
 - 2.5. Update the product release burndown chart
3. Release planning
 - 3.1. Prioritize the product backlog
 - 3.2. Estimate the product backlog
 - 3.3. Set the overall features and functionality that the release will contain
 - 3.4. Set the goals and establish a plan for the release
 - 3.5. Define major risks
 - 3.6. Define probable delivery date and cost
 - 3.7. Establish Scrum rules
4. Sprint planning
 - 4.1. Set the sprint goal, i.e. what will be done in the sprint
 - 4.2. Establish the spring backlog (list of task in the sprint)
 - 4.3. Identify tasks for the sprint
 - 4.4. Design the sprint work
 - 4.5. Define activities to achieve the sprint goal
5. The sprint
 - 5.1. Develop the product
 - 5.2. Test the product
 - 5.3. Documentation
 - 5.4. Review of estimated remaining work (by Team members)
6. Sprint review
 - 6.1. Product Owner acceptance tests the increment of the product
 - 6.2. Team suggests what to do next
 - 6.3. Review the BurnDown chart (by Product Owner)

- 7. Sprint retrospective
 - 7.1. Create a prioritized list of the major items of success in the sprint and how to improve in the next sprint
 - 7.2. Create a prioritized list of the major items of impediments for the team and how to remove them
- 8. Daily Scrum meetings
- 9. Find current risks and impediments

4.2. Plan-driven Project Management Practices

The basic project management practices of process models have been combined and described in [15] that form a part of the theoretical bases for the current study. The project management and related practices were combined there from CMMI and IS 15504. Project management activities were added from the PMBoK and from over 12 sources of project management literature. The set of basic project management activities described in [15] is viewed as the set of plan-driven project management practices in the current study.

The project management practices from process models can be viewed as best practices. They are the specific practices from the Project Planning and Project Monitoring and Control process areas of CMMI, and the Project Management base practices from IS 15504. Project management activities of the PMBoK and project management literature that are not described in the process models are also added to the set of plan-driven project management practices. The final set of plan-driven project management practices is grouped together with Scrum practice in chapter 5.3 in Table 1. The method followed in grouping these practices has been described in greater detail in [15].

4.3. Grouping the Plan-driven and ScrumProject Management Practices

The following table (Table 1) shows the grouping of the plan-driven and Scrum project management practices. A row without a corresponding Scrum practice indicates that the practice is not described in Scrum on explicit practice level and is therefore unknown to Scrum.

Table 1.Grouping the plan-driven and Scrum project management practices.

IS 15504	CMMI	Scrum
<i>Project Management</i>	<i>Project Planning, Monitoring & Control</i>	
MAN.3.BP1: Define the scope of work	SP 1.1 Estimate the scope of the project	2.1 Create and maintain PB
MAN.3.BP2: Define project life cycle	SP 1.3 Define project life cycle	3.7 Establish Scrum rules
MAN.3.BP3 Evaluate feasibility of the project		
MAN.3.BP4: Determine and maintain estimates for project attributes	SP 1.2 Establish estimates of work products and task attributes SP 1.4 Determine estimates of effort and cost	3.1 Prioritize PB 3.2 Estimate PB 4.1 Set the sprint goal 2.3 Set the acceptance criteria to items of PB
MAN.3.BP7: Define project schedule	SP 2.1 Establish project budget and schedule	3.2 Estimate PB 3.6 Define probable delivery date and cost
	SP 2.2 Identify project risks	3.5 Define major risks

MAN.3.BP5: Define project activities and tasks		4. Sprint planning
MAN.3.BP6: Define needs for experience, knowledge and skills	SP 2.5 Plan for needed knowledge and skills	
	SP 2.4 Plan for project resources	
	SP 2.3 Plan for data management	
MAN.3.BP8: Identify and monitor project interfaces		
MAN.3.BP9: Allocate responsibilities	SP 2.6 Plan stakeholder involvement	
MAN.3.BP10: Establish project plan	SP 2.7 Establish the project plan	4.1 Set the sprint goal 4.2 Establish SP 4.3 Identify tasks for the sprint
MAN.3.BP11: Implement the project plan		4.4 Design the sprint work 4.5 Define activities to achieve the sprint goal 5. The sprint
	SP 3.1 Review plans that affect the project	
	SP 3.2 Reconcile work and resource levels	
	SP 3.3 Obtain plan commitment	4.2 Establish SB
MAN.3.BP12: Monitor project attributes	SP 1.1 Monitor project planning parameters	1.1 Monitor sprint work 1.2 Schedule control 2.5 Update RBD
	SP 1.2 Monitor commitments	1.1 Monitor sprint work
	SP 1.3 Monitor project risks	8.1 Find current risks and impediments 7.2 List impediments
	SP 1.4 Monitor data management	
	SP 1.5 Monitor stakeholder involvement	
MAN.3.BP13: Review progress of the project	SP 1.6 Conduct progress reviews	6.3 Review SBD 5.4 Review of estimated remaining work
	SP 1.7 Conduct milestone reviews	6. Sprint review
	SP 2.1 Analyze issues	7.2 List impediments
		7.1 List success factors
MAN.3.BP14: Act to correct deviations	SP 2.2 Take corrective action	5. The sprint
	SP 2.3 Manage corrective action	1.3 Remove organizational impediments that impede the Scrum
<u>Project finalizing activities from the PMBoK:</u>		<u>Scrum</u>
Information to formalize project completion is gathered and disseminated		
The project is evaluated after closing		
The lessons learned are compiled for future projects		
<u>Project Directing activities from the project management literature:</u>		<u>Scrum</u>
Directions are given to the project team		Covered by ScrumMaster and daily Scrum meetings
Supervise the project team		
Motivate the people in the project team		
Coordinate the interactions between the people in the project team		
Explain the reasons behind the decision-making to the project team		
Resolve the conflicts within the project team		

According to the grouping in the table above (Table 1), the following practices were unknown to Scrum: evaluating the feasibility of the project, planning and monitoring the data management, planning for needed knowledge and skills, resource planning and stakeholder involvement planning, stakeholder involvement monitoring, reviewing plans that affect the project, reconciling work and resource levels, the project finalizing activities of the PMBoK and project directing activities from the project management literature.

Most of the plan-driven practices that are unknown to Scrum are covered either by the responsibilities of the ScrumMaster (evaluating the feasibility of the project, planning for needed knowledge and skills, resource and stakeholder involvement planning and monitoring, project directing) or in the daily Scrum meetings (reconciling work and resource levels, reviewing plans that affect the project) but they have not been explicitly described as practices.

The unknown practices of Scrum not covered by ScrumMaster responsibilities nor addressed in daily Scrum meetings are about planning and monitoring the data management described in CMMI and project finalizing described in the PMBoK. Data management includes the processes and systems that plan for, acquire and provide stewardship for business and technical data throughout the data life-cycle [9]. The practice of data management addresses the organizational level and requires rigorous planning and monitoring that is too heavy for an agile method. The practices and rules defined in Scrum contribute for good communication and promote collaboration between team and stakeholders, project information is shared in meetings or document available to everyone [4]. Scrum relies on the tacit knowledge rather than documentation [7].

At the same time, Scrum would benefit from the postmortem analysis carried out in the end of the project described in the PMBoK. The causes of failed projects should be analyzed, understood and communicated in order to improve the performance of subsequent project [13].

The only unknown practice for the plan-driven methods from Scrum is one of the practices from sprint retrospective - prioritize major items of success in the last sprint and describe how to improve in the next sprint. According to CMMI, project related issues and impediments are analyzed, corrective action is taken and monitored. Scrum suggests carrying not only the impeding but also the success factors into the next sprint to increase a chance to improve immediately. Similar practice could be added to the plan-driven practices.

4.4. Tailoring the Scrum Process

The comparison of practices described in Table 1 illustrates that Scrum is not targeting the organizational level practices. As was stated by Turner and Jain in [17] the scope of agile approach and of CMMI differs. CMMI has broad, inclusive and organizational approach while agile has small and focused approach. Sutherland et al. also found in [19] that CMMI has a concept of institutionalization that can help establish needed discipline to adopt agile methods organization wide. According to [6] the focus of the agile paradigm is on project and team while CMMI is implemented at organizational level. Therefore Scrum project management could be improved by adding practices that address organizational level or enhance the communication between project and organizational level.

The CMMI specific practices about planning the project resources and for the needed knowledge and skills are handled in the case-company during the project preparation phase but they are not addressed in Scrum rules and activities or by any Scrum roles. In addition to that, Scrum could benefit from postmortem reviews corresponding to the three unknown practices of Scrum from the PMBoK. Also, the output work products of project progress review described in process models could contribute to better managerial overview of Scrum project status during project development. All these improvements are pointing towards organizational level shortage.

CMMI [9] describes the typical work products of progress and milestone reviews as documented progress or milestone review results. IS 15504 describes the progress status record in detailed level saying that it has the following possible attributes: the status of actual tasks against planned tasks, status of actual results against established goals, status of actual resource allocation against planned resources, status of actual cost against budget estimates, status of actual time against planned schedule, status of actual quality against planned quality and record of any deviations from planned activities and reasons why (15504-5).

In order to enhance the communication between project and organization levels, the progress status reports could be added to the tailored Scrum process as described in Figure 2.

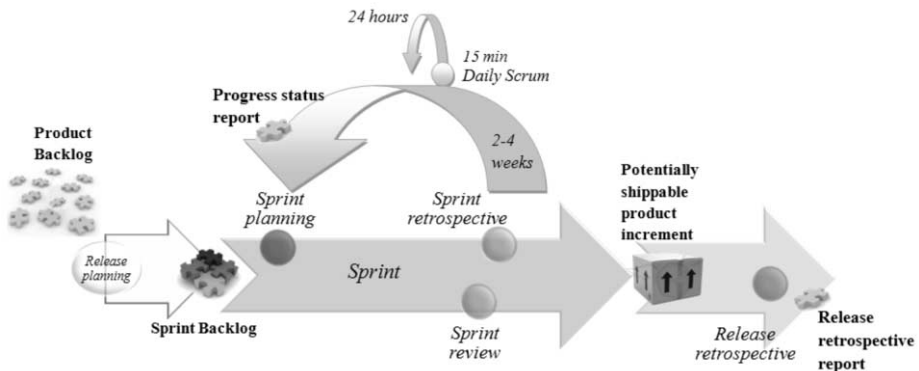


Figure2. The tailored Scrum process (*activities in Italic, artifacts in Bold*)

The progress status reports are periodical reports written by either ScrumMaster or Product Owner and distributed to the team and management. The progress status report includes the description of the status of tasks, resources, costs, schedule and quality against their estimates.

In addition to the progress status report that has been added to the tailored Scrum process, the project finalizing activities of the PMBoK unknown to Scrum are also added to support the increase in project performance in subsequent projects. The three PMBoK practices are: information to formalize project completion is gathered and disseminated, the project is evaluated after closing and the lessons learned are compiled for future projects. The new practice for the tailored process is called *release retrospective* and will include similar tasks of sprint retrospective, i.e. create a prioritized list of major items of success in the release and how to improve in subsequent releases; and create a prioritized list of major items of impediments in the release and how to remove them from subsequent release. These tasks correspond to the PMBoK practices of project evaluation and compiling the lessons learned. The

release retrospective has an output artifact of *release retrospective report* which is shared among the team, posted in the collaboration tool of the company and emailed to all ScrumMasters and Product Owners for organizational knowledge sharing purpose. The ScrumMaster and the Product Owner of the release share the responsibility for the practice to be carried out and artifact delivered. The sharing of the release retrospective report corresponds to the PMBoK practice of being gathering and disseminating information of project completion.

5. Conclusions and Future Work

In our research, we compared the plan-driven project management practices with Scrum project management practices and tailored the industry project management process so that it would address the organizational level.

Next, we plan to measure the increase of estimation accuracy and project performance of the tailored process. In order to do that, the tailored process will be implemented in industry for evaluation purposes. The data will be collected about project performance and estimation accuracy. According to Salo and Abrahamsson [12], the main problem with agile development is the lack of detailed planning of the iteration, namely the effort estimation. Project estimation accuracy should increase when the underlying causes for the gaps between the estimated and realized effort for each task have been realized. With the progress status reports and release retrospective reports the estimation accuracy should increase. The estimation accuracy will be followed after implementation of the tailored process and compared to the earlier estimation results.

Project performance will be measured through both quantitative and qualitative measures collected in industry cases. The project performance factors used in this study are described in [20] - the project's ability to meet budget commitments, ability to meet schedule commitments, ability to achieve customer satisfaction, ability to meet the defined goals, productivity in the project, and the product's ability to satisfy specified requirements. The data on budget, schedule, project and product goals are collected in the case company and will be compared to the data prior to implementing the tailored process. Customer satisfaction surveys are carried out in each project and are compared to see whether the tailored process has increased the customer satisfaction. The project productivity is measured through project velocity that is described in [12] where the increased project velocity corresponds to increased project productivity.

Acknowledgments

This research was supported by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS.

References

- [1] Stålthane T., Hanssen G.K.: The Application of ISO 9001 to Agile Software Development. In: Product-Focused Software Process Improvement 9th International Conference, PROFES 2008. LNCS, vol. 5089, pp. 371-385. Springer, Heidelberg (2009).
- [2] Abrahamsson P., Warsta J., Siponen M. T., Ronkainen J.: New Directions on Agile Methods: A Comparative Analysis. In: 25th IEEE International Conference on Software Engineering, 244p. Portland, Oregon (2003).
- [3] Dingsøyr T., Dybå T., Abrahamsson P.: A Preliminary Roadmap for Empirical Research on Agile Software Development. Agile 2008 Conference, 83-94 (2008)
- [4] Marçal, A. S. C.; Freitas, B. C. C.; Soares, F. S. F.; Belchior, A. D.: Mapping CMMI Project Management Process Areas to SCRUM Practices. (2008). www.cesar.org.br/files/file/SCRUMxCMMI_IEEE-final03.pdf Accessed 09.10.2009.
- [5] Controlled Chaos, www.controlchaos.com/old-site/ap.htm, Accessed 09.10.2009.
- [6] Glazer H., Dalton J., Anderson D., Konrad M., Shrum S.: CMMI or Agile: Why Not Embrace Both! CMU/SEI-2008-TN-003, Software Engineering Institute (2008), 41p.
- [7] Boehm B. and Turner R.: Balancing Agility and Discipline - A Guide for the Perplexed. Pearson Education, Boston (2004), 266p.
- [8] Hansen B., Rose J., Tjornehoj G.: Prescription, description, reflection: the shape of the software process improvement field. International Journal of Information Management, Vol. 24. 457-472 (2004)
- [9] CMMI for Development. CMU/SEI-2006-TR-008, ESC-TR-2006-008, Version 1.2, CMMI Product Team, Software Engineering Institute (2006), 537p.
- [10] ISO/IEC 15504-5. ISO/IEC 15504-5 Information Technology - Process Assessment - Part 5: An Exemplar Process Assessment Model, 1st edition, ISO/IEC JTC1/SC7 (2006), 162p.
- [11] Project Management Body of Knowledge: A guide to Project Management Body of Knowledge. Project Management Institute, Pennsylvania, 209p (2000)
- [12] Salo O. and Abrahamsson P.: An Iterative Improvement Process for Agile Software Development. Software Process Improvement and Practice, Vol. 12, 81-100 (2006)
- [13] Verner J. M. and Evancho W. M.: In-house Software Development: What Software Project Management Practices Lead to Success. IEEE Software (January/February), 86-93 (2002)
- [14] Tolfo C., Wazlawick R. S., Ferreira M. G. G., Forcellini F. A.: Agile Methods and Organizational Culture: Reflections about Cultural Levels. Software Process Improvement and Practice. (2009). <http://www3.interscience.wiley.com/cgi-bin/fulltext/122613722/PDFSTART>, Accessed 20.10.1009.
- [15] Lepmets M.: Evaluation of Basic Project Management Activities - Study in Software Industry. Tampere University of Technology, Publication 699, Pori, Finland (2007), 223p. <http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/68/lepmets.pdf?sequence=1>
- [16] Scrum Alliance (2009), <http://www.scrumalliance.org/>, Accessed 12.10.2009.
- [17] Turner R. and Jain A.: Agile Meets CMMI: Culture Clash or Common Cause? XP/Agile Universe 2002. In: Wells D. and Williams L. (eds.). LNCS, vol. 2418, pp. 153-165. (2002)
- [18] Järvinen P.: On Research Methods. Juvenes Print, Tampere, Finland (2001), 190p.
- [19] EC SME definition, http://ec.europa.eu/enterprise/enterprise_policy/sme_definition/index_en.htm, Accessed on 12.10.2009.
- [20] Goldenson, D. R. and Herbsleb J.: After the Appraisal: A Systematic Survey of Process Improvements, Its Benefits, and Factors that Influence Success. Technical Report CMU/SEI-95-TR-009, Software Engineering Institute (1995). 66p.

Test Points in Self-Testing

Edgars DIEBELIS, Prof. Dr. Janis BICEVSKIS

Datorikas Institūts DIVI, A.Kalniņa str. 2-7, Rīga, Latvia

Abstract. This paper is devoted to the implementation of self-testing, which is one of the smart technologies. Self-testing contains two components: full set of test cases and built-in testing mechanism (self-testing mode). The test cases have been collected since project start and they have been used in integration, acceptance and regression testing. The built-in self-testing mode provides execution of test cases and comparison of test results with saved standard values in different environments. This paper continues the approach described in article An Implementation of Self-Testing, expanding it with the concept of a test point, which allows flexible control of testing actions. Furthermore, the paper describes the first implementation of self-testing using test points.

Keywords. Testing, Smart technologies, Self-testing.

Introduction

The self-testing is one of the features of smart technologies [1]. The concept of smart technologies proposes to equip software with several built-in self-regulating mechanisms, which provide the designed software with self-management features and ability to react adequately to the changes in external environment similarly to living beings. The necessity of this feature is driven by the growing complexity of information systems and the fact that users without profound IT knowledge can hardly use such complex systems. The concept of smart technologies besides a number of significant features also includes external environment testing [2, 3], intelligent version updating [4], integration of the business model in the software [5]. The concept of smart technologies is aiming at similar goals as the concept of autonomous systems developed by IBM in 2001 [6, 7, 8]. Both concepts aim at raising software intellect by adding a set of non-functional advantages - ability to adapt to external situation, self-renewing, self-optimizing and other advantages. However, features and implementation mechanisms of both concepts differ significantly. The autonomous systems are built as universal and independent from properties of a specific system. As a rule, they function outside of a specific system and cooperate on the level of application interface. Hence, we may consider the autonomous systems being more like environmental properties than specific systems. Whereas the features of smart technologies provide a scaffolding, which is filled with functional possibilities of a specific system, thus integrating the implementation modules of smart technologies with the modules of a specific system. Therefore, further development of both concepts is highly valuable.

The first results of practical implementation of smart technologies are available. Intelligent version updating software was developed and is used in practice in a number of Latvian national-scale information systems, the largest of which, FIBU, manages budget planning and performance control in more than 400 government and local

government organisations with more than 2000 users [4]. Firstly, external environment testing [3] is used in FIBU, where the key problem is the management of operating systems and software versions for the large number of territorially distributed users. Secondly, external environment testing is employed by the Bank of Latvia in managing operations of many systems developed independently.. The use of smart technologies has proved to be effective in both cases [9]. The third instance of the use of smart technologies is the integration of a business model and an application [5]. The implementation is based on the concept of Model Driven Architecture (MDA) [10], and it is used in developing and maintaining several event-oriented systems. The use of smart technologies has been proven to be effective according to the results obtained in practical use. This study continues the research of the applicability of smart technologies in software testing.

Self-testing provides the software with a feature to test itself automatically prior to operation; it is similar to how the computer itself tests its readiness for operation when it is turned on. By turning on the computer self-testing is activated: automated tests are run to check that the required components, like hard disc, RAM, processor, video card, sound card etc, are in proper working order. If any of the components is damaged or unavailable, thus causing operation failure, the user receives notification. The purpose of self-testing is analogical to turning on the computer: prior to using the system, it is tested automatically that the system does not contain errors that hinder the use of the system.

The paper is composed as follows: To explain the essence of the self-testing approach, the first section repeats in brief the ideas on the self-testing method and deals with its modes [11, 12]. Section 2 deals with the concept and implementation of test point and Section 3 describes in brief the technical implementation of self-testing.

1. Method of Self-Testing

The main principles of self-testing are:

- Software is delivered together with the test cases used in automated self-testing;
- Regression testing of full critical functionality before every release of a version;
- Testing can be repeated in production, without impact on the production database.

As shown in [11, 12], self-testing contains two components:

- Test cases of system's critical functionality to check functions, which are substantial in using the system;
- Built-in mechanism (software component) for automated software testing (regression testing) that provides automated executing of test cases and comparing the test results with the standard values.

The defining of critical functionality and preparing tests, as a rule, is a part of requirement analysis and testing process. The implementation of self-testing requires at least partial inclusion of testing tools functionality in the designed system. The implementation of self-testing functionality results in complementing the designed system with self-testing functionality calls and a library of self-testing functions (.dll file). Certainly, the implementation of self-testing features requires additional efforts

during the development of the system. However, these efforts are justified by many advantages obtained in development and in long-term maintenance of a high quality system in particular.

The main feature of self-testing is ability to test the software at any time in any environment - development, test and production environments. While developing the mechanism of self-testing the developers may not enter the information into production database; however, they can be used in read-only mode. Hence, it is possible to implement testing in test or production environment without any impact on system use. Of course, it is useful to complement the set of tests with recent system modifications to ensure that testable critical functionality in self-testing is covered.

1.1. Self-Testing Software

The self-testing software is partly integrated in the testable system, which has several operating modes; one of them is self-testing mode when an automated execution of testing (process of testing) is available to the user. After testing, the user gets a testing report that includes the total number of tests executed, tests executed successfully, tests failed and a detailed failure description. The options provided by self-testing software are similar to the functionality of testing support tools.

1.2. Phases of System Testing

In order to ensure development of high quality software, it is recommendable to perform testing in three phases in different environments [11]:

- Development environment - in this environment the system has been developed, errors are corrected and system patches are made;
- Test environment - this environment is used for integration testing, error corrections and improvements.;
- Production environment - this environment is used by the system users. Patches and improvements are set only after obtaining successfully testing results in development and test environments.

Testing phases are described in detail in the article Self-Testing - New Approach to Software Quality Assurance [11].

1.3. Modes of Self-Testing

As shown in [11, 13], the self-testing functionality can be used in the following modes:

- Test storage mode. In this mode, new test cases are defined or existing test cases are edited/deleted. The system logs all necessary information of reading-writing and managing actions by recording them into the test storage file.
- Self-testing mode. In this mode, automated self-testing of the software is done by automatically executing the stored test cases. Test input data are read from the test file.
- Use mode. In this mode, there are no testing activities – the user simply uses the main functionality of the system.
- Demonstration mode. The demonstration mode can be used to demonstrate system's functionality. User can perform system demonstrations, by using stored uses cases in storage files.

Test points are used in the test storage, self-testing and demonstration modes, and they are described in detail in the following chapter.

2. Test Points and Their Implementation

A test point is a command upon which system testing actions are executed. To be more precise, a test point is a programming language command in the software text, prior to execution of which testing action commands are inserted. A test point ensures that particular actions and field values are saved when storing tests and that the software execution outcome is registered when tests are executed repeatedly. By using test points, it is possible to repeat the execution of system events.

As described in the sections above, the self-testing features are introduced in the tested system, namely - written by the test points, which can be introduced in the system in at least two ways:

- By altering the system software's source code. When developing the system, the developer implements in the software code also test points that register system's actions.
- The specialist who defines the business process schemes specifies the test points in the business process. In this case, the business processes and the software must be compatible, and extra resources for moving the testing actions to the software are required. For the time being, the authors do not have knowledge of any instances of application of the approach described above in practice.

When initially developing the self-testing software concept, it was planned to develop only test points that ensure the registration of data storage in the database and data selection from database events. It was important to check whether when executing repeatedly a database command (INSERT, UPDATA, SELECT, procedure or function call etc), the result saved in the database or selected from the database matches the data storing or data selecting performed in the first time.

While evolving the self-testing concept, the idea to use the test point approach to register all system events emerged. Thus, test points register not only data storing in database events or data selection from database events but also other application events (filling in fields in application form, calling application events etc). Such changes ensure that user interface and business logics are tested as well; also, this approach provided a possibility for users to use the system in the demonstration mode. Consequently, with comparatively low investments, the functionality of self-testing was increased considerably.

2.1. Implementation of Test Points

To implement test points in the tested system, it is required to determine how the system works, what is the structure and model of the developed system and what information that characterises the test example should be recorded when registering test examples. It is necessary in order to:

- Be able to develop new or use the existing test points that register the information necessary in the test example files in the tested system;

- Identify where exactly in the software code test points need to be placed to achieve that the critical functionality of the system is covered.

In the self-testing software there are implemented test points that add to the test example scenario the respective test point type object (Figure 1. Adding Self-Testing Test Point Objects to Test Example Scenario).

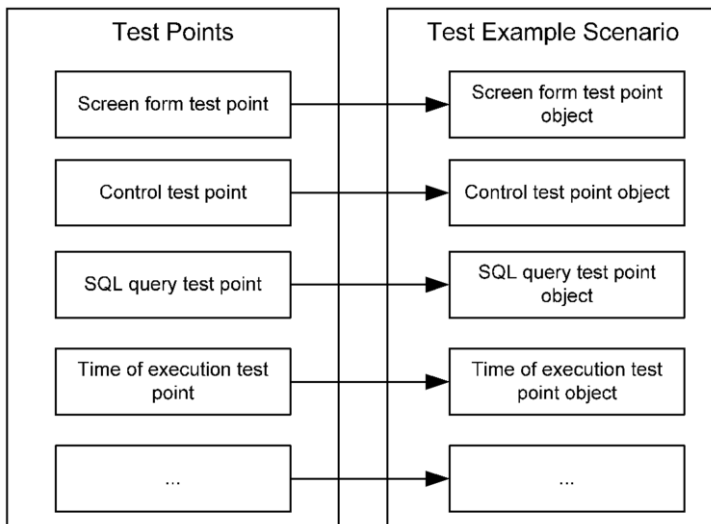


Figure 1. Adding Self-Testing Test Point Objects to Test Example Scenario

Each test point has an additional class that contains all the required information that is received by the test point. Test point objects are developed to make the work related to the test example scenario, developing the scenario and its playback easier. Currently the test point object classes specified in the figure below (Figure 2. UML Class Diagram of Test Point Objects) are implemented in the self-testing software, and they are used to register in the tested system the actions performed by the user or the system.

The self-testing software employs the following testing actions:

- Opening and closing screen forms (TestableForm). This test point registers in the test example file screen opening and closing actions. The test point type object contains the screen form name;
- Comparable value (TestableComparableField). This test point registers the values of different variables, functions and calculations that are determined in the system code and that the user does not register directly in the system. This test point is necessary to be able to register and compare the values calculated in the system. The test point can be used when the application contains a field whose value is calculated considering the values of other fields, values of which are not saved in the database. Furthermore, the test point can be used in the testing of external interfaces. The test point provides the registration of the values read from the external interface or to be delivered to it in the test example file. In the self-testing mode, operation of the external interface is

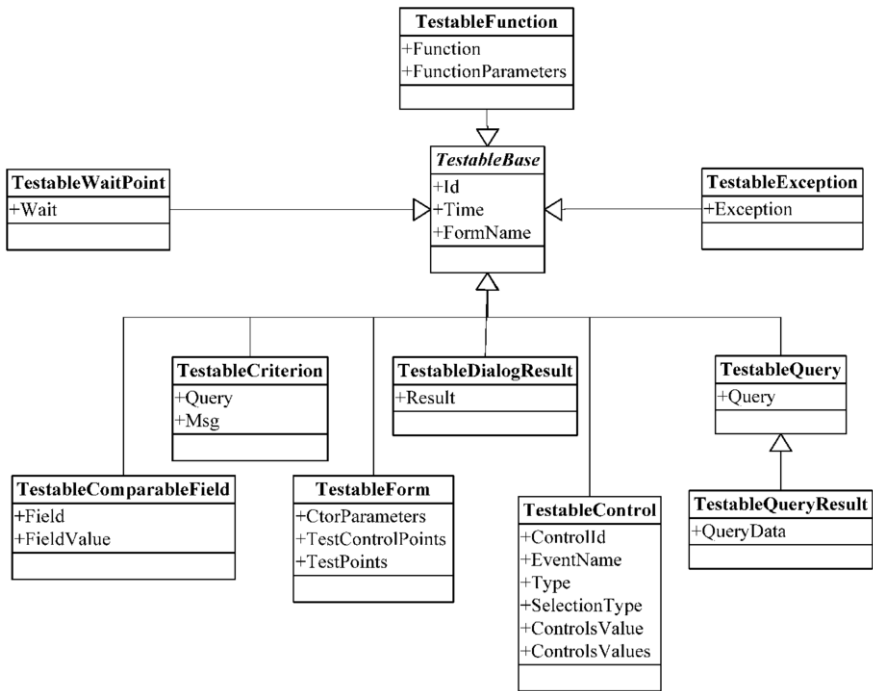


Figure 2. UML Class Diagram of Test Point Objects

simulated by reading from or delivering to the external interface the data registered in the test storage mode in the test example file;

- Test execution criterion (TestableCriterion). This test point ensures that test criteria are stored. This test point controls whether it is possible to execute the test (e.g., prior to debiting money from the account, it is checked that the account is not closed and that there is money in it). By using test execution criteria test points, it is possible to specify the criteria for the execution of the stored test. In the system self-testing mode, the test execution criteria points check whether the conditions specified in the test points are fulfilled. If the criterion is not fulfilled, the test has failed and the user can access a detailed description of test execution, in which the reason for non-execution is specified.
- Form control (TestableControl). This test point is required to register any events performed in the application, e.g. clicking on the button Save or field filling-in event.
- Dialog window (TestableDialogResult). This test point registers in the test example file the results returned by the dialog window. If the user performs actions in the dialogue window, they are registered in the test example file;
- Function (TestableFunction). This test point registers in the test example file the function call, function parameters and the result returned by the function. Executing a test that contains a function test point or a function call, the parameters delivered to the function and the result returned from the function match the information registered during test storage.

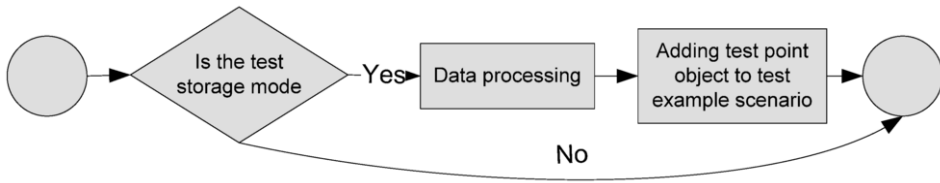


Figure 3. Test Point Operation Process

- SQL query (TestableQuery). This test point is used to register in the test file the query sent to the database;
- SQL query result (TestableQueryResult). This test point registers specific values that can be selected with an SQL query and that are compared in the test execution mode with the values selected in the test storage and registered in the test file. The SQL query test point can be used after data have been saved to compare the data saved in the database, the data saved when registering the test and the data saved when performing the test repeatedly;
- Time of execution (TestableWaitPoint). This test point ensures the waiting for the execution of time consuming system processes prior to executing further actions;
- System error processing (TestableException). This test point is required to register any system errors that have occurred during software operation or playback. Test points responsible for the creation of objects of this type are added to places where software errors are processed.

Every test point call adds in the software code the respective test point type object to the test example scenario (Figure 3. Test Point Operation Process). Unlike other test points, the screen form control test point in the test example scenario can rewrite the previous screen form control test point. If the action to be registered in the test point takes place in one control, e.g. in the test field value 'a' and then value 'b' are entered and they together create 'ab', then in the test example scenario it is registered that value 'ab' has been entered in the test field.

In every test point function that registers in the test example file the information required for the test example there is implemented a check that identifies the mode in which the system operates. If the system operates in the use or demonstration mode, then the test point functions terminate their operation immediately after calling. If the system is used in the test storage or self-testing mode, the test point functions in the test example file register the information that characterises the test example.

To ensure conveniently manageable and usable maintenance of test examples, they will be registered and stored in XML files.

2.2. Using Test Points in Modes of Self-Testing

Test points are placed by the developers in the system to achieve that the critical functionality of the system is covered. Test points are used as follows:

- Test storage mode. When the user creates a new test, the specified information in the test file and obtained in testing is registered in the test points implemented in the system. Various types of information can be registered in

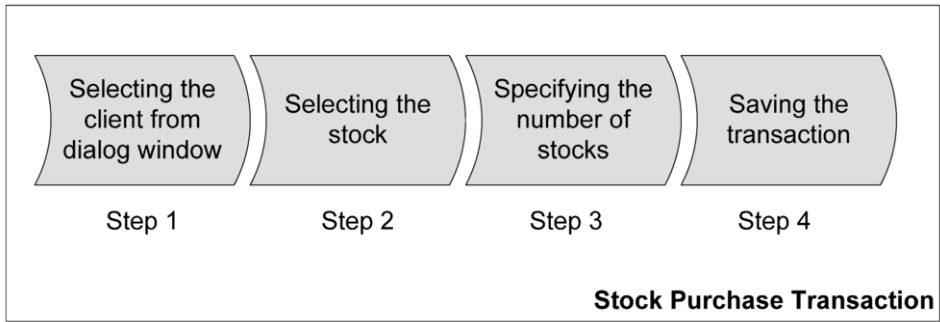


Figure 4. Stock Purchase Transaction Process

test points, e.g. value of filled-in fields, clicking a command button, selecting a value from a list etc.;

- Self-testing mode. The software automatically executes the events registered in the test files, replacing the events entered during storage with their selection from the test file. The test points placed in the system during execution of tests create the same test file as in the test storage mode. When the testing is finished, the file created in the test storage mode is compared with the test file created in the self-testing mode. If the contents of the files match, the test has been successful; if they do not match, the testing has failed;
- Demonstration mode. In the demonstration mode, the test files that have been created in the test storage mode and successfully executed in the self-testing mode are used. In the demonstration mode, within a defined time interval or when the user executes commands from the test file step by step, the functionality of the system can be demonstrated both to teach new system users and to demonstrate the system functionality to any potential its buyers.

2.3. Example of Test Point Use

To show how test points are used, a stock purchase transaction process is shown in the next figure (Figure 4. Stock Purchase Transaction Process). The registration of a stock purchase transaction consists of the following main steps:

- Specifying the client;
- Selecting the stock;
- Specifying the number of stocks;
- Saving the transaction.

To implement self-testing in the stock purchase transaction process, the system would have the following five test points, which various testing actions are written to:

1. Test point *Dialog window* registers the client selected in it in the test storage file (Step 1).
2. Test point *Form control* registers in the test storage file the stock specified for the transaction (Step 2).
3. Test point *Form control* registers in the test storage file the quantity of stocks specified for the transaction (Step 3).
4. Test point *Form control* registers in the test storage file the event of clicking on the button Save (Step 4).

5. Test point *SQL query result* registers in the test storage file the data saved in the database after clicking on the button Save (Step 4).

Classification of test points is outlined in detail below in this Section.

When a stock purchase transaction test case is registered, each of the points in the test storage file registers information that is used to play back the test. When a stock purchase transaction test is played back, the self-testing software, step by step, reads from and executes the actions registered in the test file. When the actions specified in the test file are executed, a new test file is created. When all the actions have been executed, the test files are compared; they should match if the tests have been successful. If the files do not match, the user is able to identify in the testing software application the point (command) in the test file that has been executed with errors.

3. Self-Testing Software

The self-testing software is part of the system to be developed. It means that there is no need to install additional testing tools for system testing at the system developers, customers or users. System testing is done by the self-testing software that is partially integrated in the tested system.

Key components of the self-testing software are:

- Test control block. Users use the test control block to perform various basic operations related to test registration and playback;
- Library of test actions. The library contains testing functions that register in the test file the testing actions performed;
- Test file (XML file). This file contains all the required information about the registered test example.

3.1. Test Control Block

The first version of the test control block has been developed. The test control block has been developed with additional functionality and improved with user interface.

The test control block consists of two modules.

- Test control module. The control module is responsible for the control of test examples: it makes it possible to load test examples and delivers tests for execution;
- Test playback module. The test playback module is responsible for test playbacks and for notifying test execution results.

3.1.1. Test Control Module

The control module is a program with a user interface that ensures test management and test execution and the comparing of results (Figure 5. Test Control Module).

The control module ensures simultaneous loading and execution of several test examples. Also, the control module determines the succession in which test examples are executed. Simultaneous execution of test examples provides the possibility to simulate simultaneously actions of a number of users in the testing system. The obtained data on the execution of test examples (test execution time) can be used to

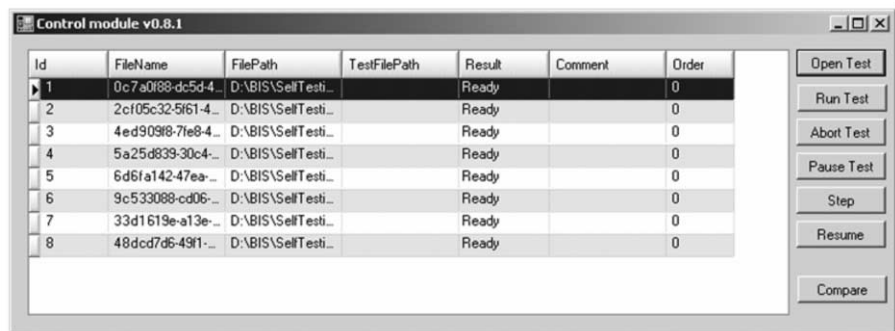


Figure 5. Test Control Module

analyse the system’s speed. Successive test example execution can be used if it is required to check several tests as one entire test.

The test control module provides the following functions:

- Delivering the test execution commands initiated in the control module to the test playback module;
- Management of test examples. Loading of test examples and delivering tests for execution;
- Defining the configuration of the self-testing software:
 - Selecting the system operation mode (test storage, self-testing, use, demonstration);
 - Selecting the test mode. The user can specify whether tests should be executed in visible or invisible mode. The visible mode is intended for demonstrations; but if the user wants, they can follow test execution step by step. The invisible mode provides for faster test execution;
 - Feature that ensures the possibility in the test storage mode to register in the test example file all the data returned from the database;
 - Feature that ensures the possibility in the self-testing mode to use the data registered in the test storage mode in the test example file. This functionality makes it possible to repeat the test example execution under the same conditions that were in place when the test was stored. The condition for the use of the feature – when the test example is registered, a feature that registers all the data returned from the database in the test example file must be selected.
- Information on test execution. If the test fails, the control block will provide the user with information on reasons for the failure;
- Deleting tests and test files.

3.1.2. Test Playback Module

The test playback module is responsible for playing black test examples. The module is independent from the test control module. It ensures continuous operation of the control module also in the cases when an error occurs in the test playback module.

The test playback module is a console program, which receives as operation parameters the test file and the program channel name. The program channel is used to ensure interactive playback of the test. Through the program channel, commands from

the control module are delivered to the test playback module. The test playback module provides the following key functions:

- Receiving commands from the control module;
- Abort test;
- Pause test. This function can be used to record a new test or to suspend the system during demonstration in order to tell the interested people on a particular system functionality in detail;
- Resume test. This operation is possible if before it the Pause Test or Step operation has been performed;
- Step. This operation stops the automatic execution of the test and ensures that the test can be played back step by step (by test point). This operation can be used in both the demonstration and self-testing modes;
- Create new from existing point. This operation registering a new test example based on an already stored test example. When this operation is executed, a new test example file is created; in its beginning a reference to the related test example file and test points to be executed is recorded.

The result from the test playback module is returned to the control module as a string that contains several elements:

- Test result: the test has been successful or failed, the related test file has not been found or an error of the self-testing software;
- Path reference to the new test file, which was created during test execution.

3.2. Library of Test Actions

The library contains the test action functions described herein. Testing action function calls are implemented in the tested system. Test functions are assigned parameters that characterise the test action. Testing functions, on the basis of the received parameters, make the respective records in the test file.

3.3. Test File

Test file (XML file). Test functions in XML file, using a particular structure, register the values that characterise the test case. The XML file structure and example are described in detail in the article An Implementation of Self-Testing [13].

4. Conclusions

In order to present advantages of self-testing, the self-testing features are integrated in a large and complex financial system. Although efforts are ongoing, the following conclusions can be drawn from the experience:

1. Introduction of a self-testing functionality is more useful in incremental development model, especially gradually developed systems and systems with long-term maintenance and less useful in the linear development model.
2. Self-testing significantly saves time required for repeated testing (regression) of the existing functionality. This is critical for large systems, where minor modifications can cause fatal errors and impact system's usability.

3. Self-testing requires additional efforts to integrate the functionality of self-testing into software, to develop critical functionality tests and testing procedures.
4. The introduction of self-testing functionality would lower maintenance costs and ensure high quality of the system.
5. Self-testing does not replace traditional testing of software; it modifies the testing process by increasing significantly the role of developer in software testing.
6. Test points make test recording and automatic execution much easier. Test points ensure that tests can be recorded in a convenient and easy-to-read manner.
7. Test execution criteria test point determines the possibility to execute the test using the available data set.
8. If test execution criteria test points are used, it is not necessary to maintain the data set which was used to register the test.
9. If test points are used, the user can, independently from the developer, register and then repeatedly execute test cases.
10. Test execution criteria test point provides a possibility to execute tests in random order.
11. The use of self-testing is simple, and system developers should not be afraid of using self-testing.

References

- [1] Bičevska, Z., Bičevskis, J.: Smart Technologies in Software Life Cycle. In: Münch, J., Abrahamsson, P. (eds.) *Product-Focused Software Process Improvement. 8th International Conference, PROFES 2007*, Riga, Latvia, July 2-4, 2007, LNCS, vol. 4589, pp. 262-272. Springer-Verlag, Berlin Heidelberg (2007).
- [2] Rauhvargers, K., Bicevskis, J.: Environment Testing Enabled Software - a Step Towards Execution Context Awareness. In: Hele-Mai Haav, Ahto Kalja (eds.) *Databases and Information Systems, Selected Papers from the 8th International Baltic Conference, IOS Press vol. 187*, pp. 169-179 (2009).
- [3] Rauhvargers, K.: On the Implementation of a Meta-data Driven Self Testing Model. In: Hruška, T., Madeyski, L., Ochodek, M. (eds.) *Software Engineering Techniques in Progress*, Brno, Czech Republic (2008).
- [4] Bičevska, Z., Bičevskis, J.: Applying of smart technologies in software development: Automated version updating. In: *Scientific Papers University of Latvia, Computer Science and Information Technologies*, vol. 733, ISSN 1407-2157, pp. 24-37 (2008).
- [5] Ceriņa-Bērziņa J., Bičevskis J., Karnītis G.: Information systems development based on visual Domain Specific Language BiLingva. In: *Preprint of the Proceedings of the 4th IFIP TC 2 Central and East Europe Conference on Software Engineering Techniques, CEE-SET 2009*, Krakow, Poland, Oktober 12-14, 2009, pp. 128-137.
- [6] Ganek, A. G., Corbi, T. A.: The dawning of the autonomic computing era. In: *IBM Systems Journal*, vol. 42, no. 1, pp. 5-18 (2003).
- [7] Sterritt, R., Bustard, D.: Towards an autonomic computing environment. In: *14th International Workshop on Database and Expert Systems Applications (DEXA 2003)*, 2003. Proceedings, pp. 694 - 698 (2003).
- [8] Lightstone, S.: Foundations of Autonomic Computing Development. In: *Proceedings of the Fourth IEEE international Workshop on Engineering of Autonomic and Autonomous Systems*, pp. 163-171 (2007).
- [9] Bicevska, Z.: Applying Smart Technologies: Evaluation of Effectiveness. In: *Conference Proceedings of the 2nd International Multi-Conference on Engineering and Technological Innovation (IMETI 2009)*, Orlando, Florida, USA, July 10-13, 2009.
- [10] J. Barzdins, A. Zarins, K. Cerans, M. Grasmanis, A. Kalnins, E. Rencis, L.Lace, R. Liepins, A. Sprogis, A.Zarins.: Domain Specific languages for Business Process Managment: a Case Study Proceedings of DSM'09 Workshop of OOPSLA 2009, Orlando, USA.

- [11] Diebelis, E., Takeris, V., Bičevskis, J.: Self-testing - new approach to software quality assurance. In: Proceedings of the 13th East-European Conference on Advances in Databases and Information Systems (ADBIS 2009), pp. 62-77. Riga, Latvia, September 7-10, 2009.
- [12] Bičevska, Z., Bičevskis, J.: Applying Self-Testing: Advantages and Limitations. In: Hele-Mai Haav, Ahto Kalja (eds.) Databases and Information Systems, Selected Papers from the 8th International Baltic Conference, IOS Press vol. 187, pp. 192-202 (2009).
- [13] Diebelis, E., Bičevskis, J.: An Implementation of Self-Testing. In: Proceedings of the 9th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2010), pp. 487-502. Riga, Latvia, July 5-7, 2010.

This page intentionally left blank

Information Systems and Security

This page intentionally left blank

Integrating IT Governance, Risk, and Compliance Management Processes

Nicolas RACZ,^{a,1} Edgar WEIPPL^a and Andreas SEUFERT^b

^a*TU Vienna*

^b*Steinbeis Hochschule Berlin*

Abstract. Even though the field of Governance, Risk, and Compliance (GRC) has witnessed increased attention over the last years, there is a lack of research on the integrated approach to GRC. This research suggests an integrated process model for high-level IT GRC management. After a discussion of existing process models for integrated GRC, the scope of the research within GRC is defined and explained. Frameworks for the separate topics of IT governance, IT risk management, and IT compliance management are selected and discussed. Finally these frameworks are merged into a single integrated process model. The model is then validated through a comparison to IT GRC processes of three multinational companies.

Keywords. integrated, IT GRC, governance, risk management, compliance, process model, information technology

1. Motivation

The term “GRC” was created by PricewaterhouseCoopers in 2004 [1] mainly as a response to a number of important regulations, such as the Sarbanes-Oxley-Act. While the industry has been quick in adopting the term promoted by auditing and consulting services and software to manage GRC processes, a recent study remarked on the lack of scientific research on the integrated approach to governance, risk management, and compliance [2]. While there is a lot of research on the three disciplines as separate topics, there is little discussion on how potential synergies can be leveraged. For information systems research the field of integrated GRC is interesting from two main perspectives [3]. Firstly as an instrument; how can information systems support integrated GRC in an organisation’s (business) operations – what is “IT for GRC”? And secondly with IT as the subject matter – what is “GRC for IT” or “IT GRC” [4], i.e. how can integrated GRC be applied to an organisation’s information technology landscape? IT GRC is best understood as a subset of GRC that supports IT operations in the same way as GRC as a whole supports business operations. It is aligned with IT operations and with an organisation’s overall GRC strategy. The integration of IT governance, IT risk management and IT compliance has not yet been adequately researched. As more than half of GRC publications primarily treat software technology [1], it may be assumed that major integration opportunities exist on a technology level. But before technology can be examined, first the IT GRC processes that are supported should be investigated.

¹ Corresponding Author: Nicolas Racz, TU Vienna, Austria; E-mail: info@grc-resource.com.

The goal of this research paper is to describe a process that helps to enable the management of IT governance, risk management and compliance in an integrated manner, allowing technology decisions to be made and risks and compliance activities to be managed based on an all-encompassing perspective of IT GRC.

2. Prior Research

There are five frameworks of differing detail that claim to integrate GRC. Mitchell proposes a framework to drive “principled performance” [5]. According to this model, an enterprise tries to overcome obstacles and achieve its objectives while staying between mandated and voluntary boundaries. Mitchell lists ten areas that are part of GRC and that share a common meta-process: objective setting, boundary identification, risk assessment, proactive actions, detection and checking, response, evaluation, improvement, and communication. While delivering this notable insight, he does not go further and lay out in detail how these common processes could be shared across the ten areas to leverage synergies.

The same author was also involved in the creation of the “GRC Capability Model” of the Open Compliance and Ethics Group (OCEG), an exhaustive model consisting of nine components (categories) and 29 sub-elements, for each of which core sub-practices are listed [6]. The OCEG model is certainly very useful for professionals who want to gain an understanding of all possible GRC activities. However it does not distinguish between operative and management processes. Furthermore, it does not explicitly point out where the integration of formerly distinct disciplines takes place. Sometimes the adopted integration can be guessed, but this is easy only in obvious cases, such as when compliance risks are mentioned in risk analysis. Moreover the model shows only few governance aspects in the GRC process. While the role of governance is explained in detail in the introductory sections, in the process model it only reappears in the sub-practice “analyze governance culture and management style.” The extent to which the model supports governance processes is not clarified. Lastly, due to its greenfield development approach the model does not explain how it relates to existing standards.

Sachar Paulus provides a “GRC reference architecture” [7] consisting of four major phases: requirements modeling, status investigation, situation improvement, and crisis and incident management. While this model is concise and easy to understand, it contradicts the generally more common comprehensive understanding of GRC as it claims that certain processes, such as financial risk management, do not belong to GRC. Like the OCEG model, it does a poor job in drawing out where integration between the three disciplines is accomplished. The framework described by Frigo and Anderson [8] lacks detail and arbitrarily mixes processes with organisational entities and objectives. Tapscott [9] suggests an integrated approach to GRC based on four core values, but he does not translate this approach into a process model.

Unfortunately none of the five models explicitly treats IT GRC, leaving their applicability to GRC for information technology in question. As the derivation of the models from existing standards, research, or best practices is also hardly visible, we conclude that a scientific process model for integrated IT GRC has yet to be created.

3. Research Methodology

The methodology applied in this research consists of four stages depicted in Figure 1:

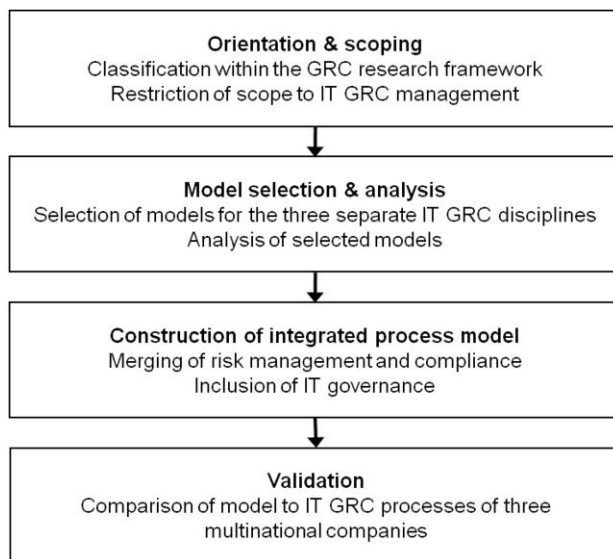


Figure 1. Research methodology

The complexity of the governance, risk and compliance domain mandated a clear classification of the elements of GRC to be considered in our research. Based on a previously developed GRC definition and frame of reference for GRC research, we restricted the scope to IT GRC management. Consequently we selected and analyzed models for the three separate IT GRC disciplines – IT governance, IT risk management, and IT compliance. Then we merged the three selected high-level process frameworks into a single process model. First we explained how IT compliance can be integrated with risk management through consideration of the risk of non-compliance and the mapping of IT compliance processes to similar processes in risk management. Second we examined the relation of IT governance to IT risk management and IT compliance before merging the three disciplines in a single process model through identification of commonalities and mapping of overlapping or combinable processes. In the final stage of the research a rough validation was carried out through the comparison of the model with IT GRC processes in three multinational companies.

4. Towards a Process Model for Integrated IT GRC Management

In the following we describe the selected process models for IT governance, (IT) risk management, and IT compliance.

4.1. Orientation and Scoping

For scoping and orientation within the GRC domain we built on a definition of GRC that we had developed in previous research [1]. At present this definition is the only scientifically derived and validated GRC definition:

GRC is an integrated, holistic approach to organisation-wide governance, risk, and compliance ensuring that an organisation acts ethically correct and in accordance with its risk appetite, internal policies, and external regulations through the alignment of strategy, processes, technology, and people, thereby improving efficiency and effectiveness.

The definition was translated into a frame of reference for GRC research that incorporates the disciplines, rules, components, characteristics and objectives of GRC (Figure 2). The research at hand demonstrates the integration of high-level processes of the three disciplines of IT governance, IT risk management, and IT compliance. Hence the scope of operations managed and supported through GRC is restricted to IT operations. The frame’s elements considered in this paper are highlighted in grey.

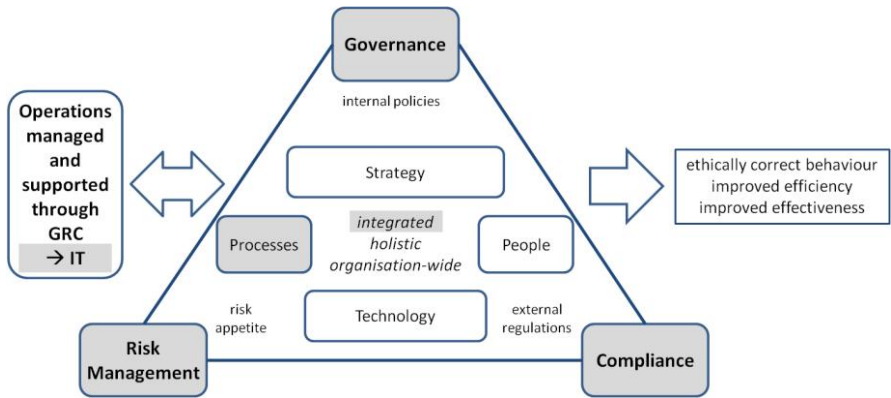


Figure 2. Elements in focus in the frame of reference for GRC research [1]

As previously mentioned, IT GRC is seen as a subset of GRC in general. The three IT GRC disciplines are subsets of their corporate counterparts as depicted in Figure 3.

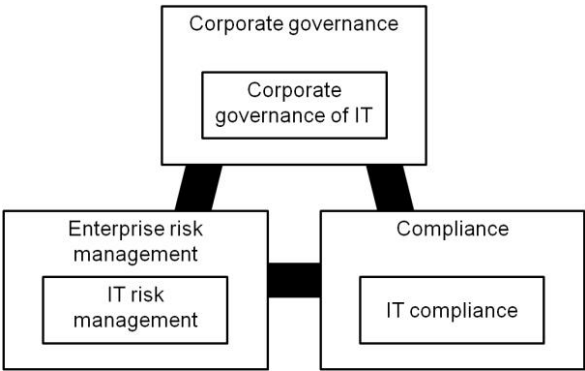


Figure 3. IT GRC as a subset of GRC

IT GRC processes were put into an enterprise process context following the new St. Gallen Management Model [10] that distinguishes three process categories: management processes, business processes, and support processes. Risk management, information management, communication, and legal processes such as compliance management are classified as support processes. Governance processes, however, are considered management processes; since they govern the allocation of resources, they belong to the operative management process category of the St. Gallen model. Consequently an IT GRC process has to be able to

- (i) support management processes through the provision of information about IT risks and IT compliance aspects and through an IT governance framework that can be referred to when taking decisions; and
- (ii) help business processes and other support processes to be executed in an effective and efficient manner through consideration of IT risks, IT compliance aspects and through an IT governance framework.

To avoid getting lost in detail we further restricted the scope to the highest level of process descriptions provided in the frameworks we included. Such a management process may be executed on a regular or an ad-hoc basis. We do not want to focus on the operative details of IT GRC, such as how an incident is stored, but on the high-level processes that provide the support outlined above.

4.2. Model Selection and Analysis

After defining the scope we analysed a variety of standards and frameworks that describe the separate disciplines of IT governance, risk management, and compliance before selecting one for each discipline as a process foundation for our research.

IT Governance

Thus far no single theory adequately explains governance in full [11]. Previous studies have focused primarily on structural mechanisms of IT governance while neglecting the process mechanisms [12]. Lewis & Millar identified two schools of thought of IT governance – one focuses on decision making and accountability, while the other is primarily concerned with controls and risk management [13]. Due to our focus on processes and the integration with risk management and compliance, we adopted the second perspective as used in the standard ISO/IEC 38500:2008 [14], describing the corporate governance of IT as

...the system by which the current and future use of IT is directed and controlled. [It] involves evaluating and directing the plans for the use of IT to support the organisation and monitoring this use to achieve plans. It includes the strategy and policies for using IT within an organisation.

We chose the process model of ISO/IEC 38500:2008 – Corporate governance of IT – over the widely implemented frameworks COBIT [15] and ITIL [16] because both of these frameworks go much further than governance through inclusion of specific practices and implementation advice for IT management, controls, and assurance. They include many aspects of risk management and compliance, which would make a clear analysis of GRC and its integration capabilities difficult. Therefore they are both more suitable for consideration at a later stage of GRC research when we

are ready to examine the extent to which these models already include integrative aspects of the process model to be developed in this paper.

Governance is not a one-and-done process, but a system that comprises processes that may be executed whenever needed. The ISO/IEC standard recommends three process steps – evaluate, direct, and monitor – to be applied across six principles: responsibility, strategy, acquisition, performance, conformance, and human behaviour. The framework putting corporate governance in context with business processes is drawn out in Figure 4.

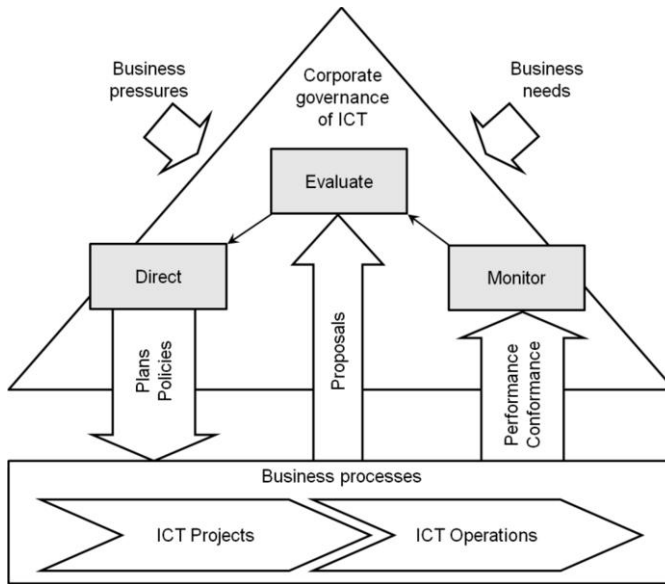


Figure 4. ISO/IEC 38500:2008 model for IT governance [14]

Ohki et al. [17] recommended adding “reporting to stakeholders” as a fourth process step, which we will consider later on.

IT Risk Management

For risk management we decided not to use a specific IT risk management model, but the more general COSO ERM framework [18]. The recently published Risk IT Framework [19] provides a model that – for our purposes – goes too far into the operative details of risk management (such as data collection) while only distinguishing the three processes risk governance, risk evaluation, and risk response at the highest process level. ISO/IEC 27005:2008 [20] focuses on information security risk management; while ISO 31000:2009 [21] (superseding AS/NZS 4360:2004 [22]) for risk management uses different wording than COSO ERM for the main processes, it basically contains the same elements. Due to risk management’s core function in GRC we assume that the selection of an enterprise risk management framework that does not focus on IT will facilitate integration with non-IT GRC in future research. COSO ERM is high-level guidance as far as IT is concerned, but specifics of IT risk management may still be considered at lower process levels [23]. The main reason to build our research on COSO ERM is that it is an enhancement of the COSO framework for

internal control [24]. This framework published in 1992 is a de-facto standard explicitly acknowledged by the US Public Company Accounting Oversight Board in its Auditing Standard No. 5 for financial reporting [25], which is referenced in the Sarbanes Oxley Act of 2002. Therefore many companies are already using the predecessor framework of COSO ERM, also for internal controls over IT [26].

According to COSO ERM “enterprise risk management” is defined as

...a process, effected by an entity’s board of directors, management and other personnel, applied in strategy setting and across the enterprise, designed to identify potential events that may affect the entity, and manage risk to be within its risk appetite, to provide reasonable assurance regarding the achievement of entity objectives. [18]

The definition was applied in the development of the COSO ERM framework.

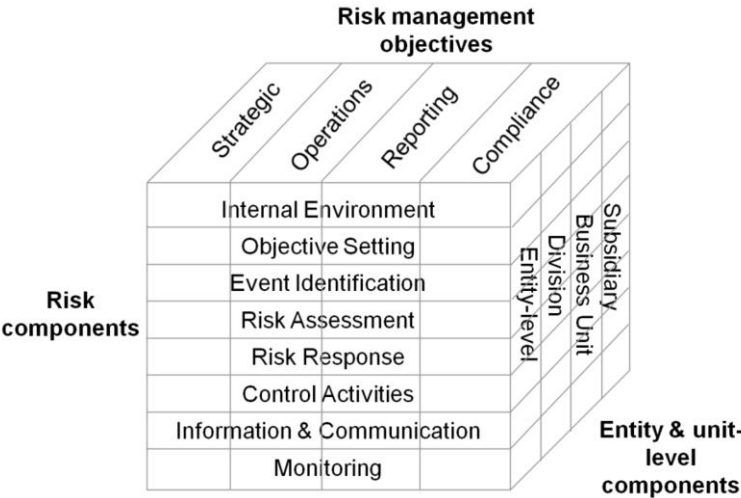


Figure 5. The COSO ERM framework [18]

The COSO cube in Figure 5 describes eight high-level processes (risk components) for risk management that are executed across the organisational hierarchy and that support the achievement of objectives in four categories. Just like governance, risk management processes are executed at varying frequencies. They might be carried out at fix time intervals, be event-driven (e.g. due to a new project or a major change in the organisation’s environment) or even be perpetual through continuous monitoring and adaptation.

IT Compliance

For IT compliance we selected the process model suggested by Rath and Sponholz [27] because this generic model can also be applied to non-IT compliance. The model divides the general process of IT compliance into four sub-processes depicted in Figure 6: requirements analysis, deviation analysis, deficiency management, and reporting/documentation.

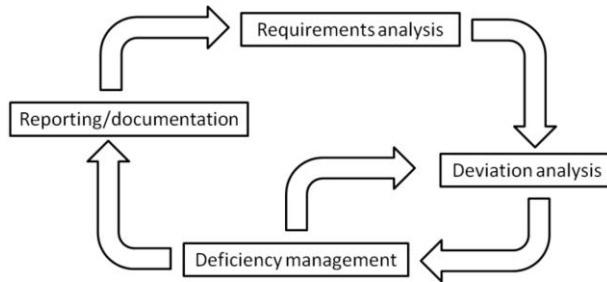


Figure 6. The IT compliance process [20]

Requirements analysis comprises the identification of regulatory, legal, contractual, and other obligations that affect the organisation's IT operations. Internal policies, such as best practices for software engineering or security guidelines, can also be included. Companies often build regulatory databases or use services such as the Unified Compliance Framework (UCF) to comprehensively collect their IT compliance requirements. The requirements build the foundation of a company's internal control system as far as IT is concerned.

Once the requirements have been identified, adherence is examined for instance through internal and external audits, self assessments, and security checks. The frequency of these examinations depends on external requirements and on the impact of potential deviations. Whereas a yearly examination will be sufficient in many cases, continuous monitoring may be recommendable in other cases.

The results of the deviation analysis define the requirements for deficiency management. At this stage existing deficiencies are eliminated through improvement of existing controls, creation of new controls, or through a makeover of parts of the control system.

All actions taken in the first three stages are documented, and relevant information is reported to internal and external stakeholders. The information reported may include incidents, sign-off status, dashboards monitoring the status of compliance activities, or key risk indicators, for instance.

4.3. Construction of a Process Model for Integrated IT GRC Management

The main commonality of the process models for separate governance, risk management, and compliance management is that they all follow a scheme that is similar to methodologies such as the PDCA cycle [28] and Six Sigma [29]. After a phase of target setting and requirements analysis action plans are defined and executed. Meanwhile monitoring ensures the proper execution of these actions and reporting informs stakeholders about the performance of the process. Improvements are implemented gradually. Now the question is: how can these process steps be defined so that governance, risk management and compliance are included comprehensively?

The most obvious way is to start from the most detailed process model we reviewed – COSO ERM – and see if it could also include governance and compliance processes. Starting from risk management also makes sense because it enables applying a risk-based perspective to governance and compliance, allowing for a quantifiable frame for decisions. We will first merge compliance management with risk management before adding governance.

Merging Risk and Compliance Management

The COSO framework states that “enterprise risk management can be expected to provide reasonable assurance of achieving objectives relating to the reliability of reporting, and compliance with laws and regulations.” [18] Hence COSO ERM already considers compliance with external laws and regulations and integrates it with its objectives categories for compliance and reporting. Internal policies or contractual compliance obligations are mentioned only in the internal environment component, but can also be added to the compliance objectives if more formalisation is required. “Reporting objectives” also includes compliance reporting, and “information and communication” asks for transparent provision of compliance information to appropriate personnel. Furthermore, COSO ERM mentions the compliance responsibilities of directors, managers, risk and financial officers, internal auditors, as well as other personnel and external parties.

Unfortunately COSO ERM does not completely integrate compliance processes with risk management processes. For example the risk of non-compliance is not mentioned in the process descriptions of event identification, risk assessment, risk response, or in the control activities. However the common procedure to join risk management and compliance on a process level is to include compliance as the “risk of non-compliance” in the risk management process. This enables a risk-based approach to compliance management – a quantitative analysis and prioritisation of actions depending on the probability and impact of a compliance violation. Merging IT compliance with COSO ERM is explained in Table 1:

Table 1. Integration of IT compliance with risk management

COSO ERM component	IT compliance component	IT compliance integration
Internal environment	Requirements analysis	Evaluation of the IT organisation’s internal environment and its congruence with the company’s overall internal environment
Objective setting	Requirements analysis	Derivation of IT compliance and IT compliance reporting objectives from business objectives
Event identification	Requirements analysis	Identification of events compromising IT compliance
Risk assessment	Deviation analysis	Assessment of the risk of non-compliance in the IT environment
Risk response	Deficiency management	Definition of how to deal with the risk of non-compliance in IT
Control activities	Deficiency management	Definition of policies and procedures to control the risk of non-compliance in IT
Information & communication	Reporting/ documentation	Definition and implementation of IT compliance reporting
Monitoring	Deviation analysis	Continuous monitoring and audits of IT compliance

IT compliance can thus be integrated and even consolidated with risk management. The degree of consolidation is defined by the joint process execution. Weak consolidation means that the risk management process is executed for the IT implementation of a certain business process, and then the whole process is repeated with regard to only IT compliance risks – often spurred by the separation of

responsibilities in an organisation. Strong consolidation means that when the risk management process is executed all IT risks including the risk of non-compliance are immediately considered as well.

Adding IT Governance

Governance in general and IT governance in the case of IT GRC represents a higher control level than risk management and compliance processes if the organisation is regarded as a cybernetic system [13]. The relationship of IT governance and IT risk management / IT compliance is twofold.

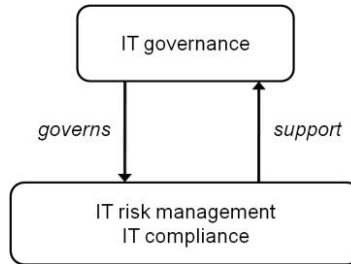


Figure 7. The relationship of IT governance to IT risk management and IT compliance

As Figure 7 shows, on the one hand risk management and compliance activities support the assertion of governance. Compliance management leads to conformance obligations (regulatory, legislation, common law, and contractual) concerning the acceptable use of IT [14]. The IT governance principle of conformance deals with evaluating, directing and monitoring compliance. ISO/IEC38500:2800 requires that risk management be applied to processes to enable conformance, and that it is applied to IT use in general and in IT acquisitions; it requires evaluating risks to continued operation, integrity of information and protection of IT assets, and it demands that risks may be reported by anyone at anytime [14].

On the other hand, IT governance governs IT risk management and IT compliance activities, i.e. it ensures that they are carried out correctly and in accordance with the ideas provided through the organisation's overall governance and IT governance. The ISO/IEC standard requires that "directors should ensure that IT used are subject to appropriate risk assessment and evaluation, as described in relevant international and national standards." [14]

IT governance provides the frame for IT risk management and IT compliance decisions. IT risk management and IT compliance management are means to help governance permeate IT operations. Respecting these relations and mapping the IT governance processes (extended with reporting) to the risk management processes with integrated compliance processes as described above, we can derive a process model for integrated IT GRC management.

The process model has the characteristics of operative management and support processes as stated in the new St. Gallen Management model. Firstly, management and business and support processes within and outside of IT operations can leverage the output of IT risk management and compliance processes. Secondly, IT governance acts as a frame of reference for IT operations, including IT risk management and IT compliance (Figure 8).

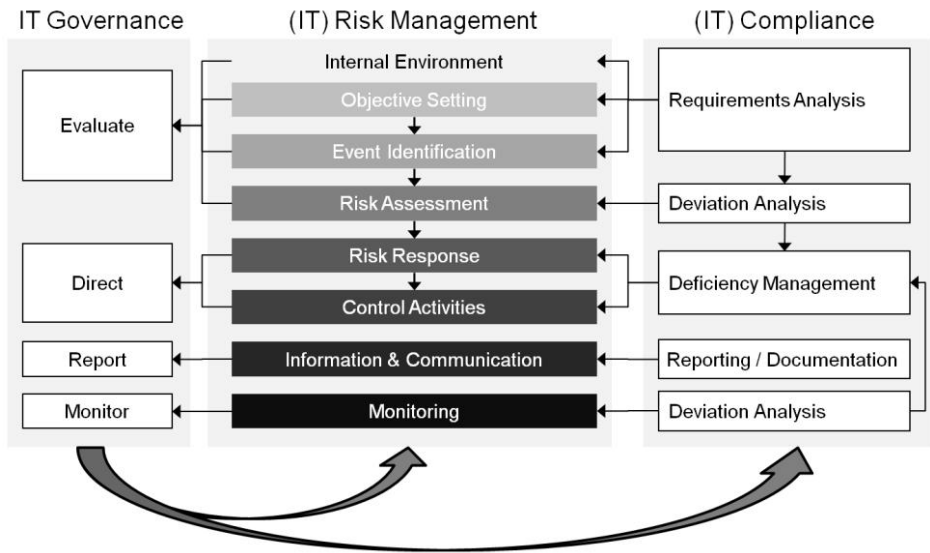


Figure 8. Process model for integrated IT GRC management [30]

A concrete example helps understand the interaction of the disciplines in the integrated model. During a periodic risk management exercise that includes IT compliance requirements analysis an IT manager finds that in one of the company's smaller markets a new national standard for data security has been introduced that surpasses the requirements of the existing standard. The manager needs to ponder investing resources to adhere to the new standard; the acquisition of costly hardware and the introduction of tedious security processes would be required. However the manager might also opt to keep the status quo, as the measures required in the new standard might increase data security only marginally compared to the investment needed to comply with the standard. The deviation analysis as part of the risk assessment process identifies several gaps; therefore the probability of non-compliance is high. In contrast the risk of a data leak is low, as existing security measures have already proven to be effective. The potential financial impact is calculated through quantitative risk assessment methods considering the loss of revenue as a consequence of reputational risk, and the ad-hoc setup of a compliance project under time pressure. From a financial point of view adherence to the standard does not seem to pay off. However the organisation's IT governance codex highlights the organisation's role as a leader in data security practices, which has been considered in the setting of compliance objectives. In order to solve the contradiction, the manager informs his boss, the company's chief information officer, of the situation, delivering the financial analysis as well as the soft facts surrounding the issue. After sitting together with managers from business, the CIO directs that the IT governance codex be revised to find out if a stronger emphasis on financial considerations in compliance decisions is needed, and that the new data security standard shall be ignored for the time being. Thus risk acceptance is chosen as risk response. The only control / deficiency management activity introduced is the monitoring of the standard's acceptance in the industry. If competitors gained an advantage through complying with the standard, the company's decision might have to be reconsidered as the financial risk might prove to

be higher than originally reckoned. Finally, the whole decision process and its results are documented and reported to the stakeholders that are concerned of governance, risk, and compliance.

4.4. Model Validation

Results from a related research can be used to validate the process model. The IT GRC management processes of three multinational companies were analyzed with a focus on integration aspects. The companies came from three different industries: software, healthcare and energy.

For validation purposes we first checked in how far the separate processes of IT governance, IT risk management and IT compliance proposed in the model coincide with the processes observed in the companies. The IT risk management process is strongly formalized in all three enterprises; on a high level the process steps match those of our process model. The IT compliance processes also look like the four steps in our model, even though IT compliance is rather described from a requirements perspective than from a process perspective. IT governance processes, however, differ in the three companies, and none of the companies considered ISO/IEC 38500:2008 when building their own frameworks. The healthcare company does not formally define IT governance at all. The software company has deployed a three-step process of identifying requirements, reviewing and approving of the proposal for the IT governance framework, and updating the framework. This requirements-oriented view of IT governance leaves out many of the ISO/IEC contents. The enterprise from the energy sector, in contrast, describes all IT governance processes used in our model, even though the wording is different and its view of IT governance surpasses the scope of our model, also including some management aspects. The ISO/IEC standard used in our model does not contradict reality in the companies; it covers most of the IT governance in the energy company, it could establish formalization of IT governance in the healthcare company, and it suggests a wider scope of IT governance for the software enterprise. Thus as used in our model each of the three disciplines on its own as used should be applicable.

Integration aspects of IT compliance with IT risk management could be observed as described in this document. The energy company carries out all IT compliance processes embedded into IT risk management. The other two companies also integrate the two activities to a certain extent through the risk of non-compliance and through risk-based audits, among other things. The integration with IT governance, in contrast, is hardly ever formalized. New material risks, changes in existing risks, and new compliance requirements trigger updates of the software company's IT governance framework. The energy company mainly links IT governance with the other functions through organisational means by centralizing responsibilities for IT risk management and IT compliance within the CIO office. Even though more formalised integration could not be observed, the general setting in the companies would enable it. The claim of the process model's relevance in practice and of its validity can be sustained.

5. Discussion

The integration of IT governance as described in ISO 38500:2008 extended with reporting, of COSO ERM and of IT compliance is feasible on a high level. It is

necessary to mention several points of critique that can be directed at our research methodology. Firstly, the selection of ISO/IEC 38500:2008, COSO ERM and the IT compliance process model might seem arbitrary to a certain extent. However all three are valid models applied in and derived from best practices in governance, risk management and compliance. Secondly, through adopting a process perspective we have disregarded structural elements of GRC and especially of IT governance. They are not indispensable in creating a process model, but they will have to be examined at another point in time to complete the governance picture. Thirdly, the applicability of the theoretic model has not yet been proven in practice. This will be done at a later stage of research once the model has been broken down to lower process levels and amended with the strategy, organisation and technology aspects of IT GRC.

The model proposed in this paper exceeds the existing GRC models presented in the prior research section in several ways, thereby extending the knowledge base of information systems research [31]. Firstly integrated GRC models have thus far been set up without showing the path from separate disciplines to an integrated approach. This is a gap from both research and professional perspectives. For research the logical deduction from the existing knowledge base is missing. For professionals it complicates change management. This gap should be filled through the approach applied in the construction of our model. Secondly our model was explicitly developed for IT GRC, whereas the applicability of existing models to the information technology domain is not evident. At the same time we did not disregard the relation of IT GRC and overall GRC. We tried to facilitate the convergence of GRC with IT GRC through selection of COSO ERM and a generic IT compliance model. Thirdly the role of governance in existing models was rather vague, whereas our model points out the two-fold relation of governance to the two other components and shows how they integrate. Finally our model is based on existing standards and best practices, and the research methodology builds on a scientifically developed GRC definition and a frame of reference for research, whereas existing models were either created in greenfield approaches or there is no explanation as to how they were derived.

6. Conclusion and Future Research

The research at hand introduces a high-level process model for integrated IT governance, risk, and compliance management, thus providing an artefact for the information systems research knowledge base. As a side effect the frame of reference for GRC research was used for the first time; its application successfully helped set and visualise the scope of the research project. It was exemplarily explained how the processes of the separate disciplines of IT governance, IT risk management, and IT compliance relate and how they can be integrated. The model's validity is given.

In future research, the processes will be broken down in order to describe the integration capabilities in more detail. We have shown that IT GRC processes have commonalities that enable integration. The integration of processes often goes hand in hand with the exploitation of synergies at the technology level. Therefore we will also examine in how far GRC technology supports the integrated IT GRC management process, how technological synergies can be leveraged, and in how far existing state-of-the-art GRC software supports the processes and integration aspects of our process model.

References

- [1] PricewaterhouseCoopers, *Integrity-Driven Performance. A New Strategy for Success Through Integrated Governance, Risk and Compliance Management*. <http://www.globalcompliance.com/pdf/PwCIntegrityDrivenPerformance.pdf> (2004)
- [2] N. Racz, E. Weippl, A. Seufert, A frame of reference for research of integrated governance, risk, and compliance (GRC), B. De Decker, I. Schaumüller-Bichl (eds.), *Communications and Multimedia Security*, 11th IFIP TC 6/TC 11 International Conference, CMS 2010 Proceedings, Springer, Berlin, 2010, 106–117.
- [3] A. Teubner, T. Feller, Informationstechnologie, Governance und Compliance, *Wirtschaftsinformatik* **50:5** (2008), 400–407.
- [4] IT Policy Compliance Group, *2008 Annual Report. IT Governance, Risk, and Compliance*, <http://www.itpolicycompliance.com/pdfs/ITPCGAnnualReport2008.pdf>
- [5] S.L. Mitchell, GRC360: A framework to help organisations drive principled performance, *International Journal of Disclosure and Governance* **4:4** (2007), 279–296.
- [6] OCEG, *GRC Capability Model. “Red Book” 2.0*, <http://www.oceg.org> (2009)
- [7] S.Paulus, *A GRC reference architecture. Overview report*. http://www.kuppingercole.com/report/sp_overview_repo_grc_arch_051009 (2009)
- [8] M.L. Frigo, R.J. Anderson, A Strategic Framework for Governance, Risk, and Compliance, *Strategic Finance* **44:1** (2009), 20–61.
- [9] D. Tapscott, *Trust and Competitive Advantage: An Integrated Approach to Governance, Risk & Compliance*. <http://www.findwhitepapers.com/whitepaper1714/> (2006)
- [10] J. Rüegg-Sturm, *Das neue St. Galler Management-Modell*, Haupt, Bern, 2003.
- [11] S.K. McGinnis, K. Pumphrey, K. Trimmer, C. Wiggins, Sustaining and extending organisation strategy via information technology governance, *Proceedings of the 37th Hawaii International Conference on System Sciences* (2004)
- [12] P.M.A. Ribbers, R.R. Peterson, M.M. Parker, Designing information technology governance processes: diagnosing contemporary practices and competing theories, *Proceedings of the 35th Hawaii International Conference on System Sciences* (2002)
- [13] E. Lewis, G. Millar, The viable governance model – a theoretical model for the governance of IT, *Proceedings of the 42nd Hawaii International Conference on System Sciences* (2009)
- [14] ISO/IEC 38500:2008, *Corporate governance of information technology*, ISO/IEC.
- [15] IT Governance Institute, *COBIT 4.1*, ISACA, Rolling Meadows, 2007.
- [16] OGC, *ITIL v3*, <http://www.itil-officialsite.com> (2007)
- [17] E. Ohki, Y. Harada, S. Kawaguchi, T. Shiozaki, T. Kagawa, Information Security Governance Framework, *Proceedings of the first ACM workshop on Information security governance* (2009)
- [18] COSO, *Enterprise risk management framework*, www.coso.org (2004)
- [19] ISACA, *The Risk IT Framework*, ISACA, Rolling Meadows, 2009.
- [20] ISO/IEC 27005:2008, *Information security risk management*, ISO/IEC.
- [21] ISO 31000:2009, *Risk management – Principles and guidelines*, ISO.
- [22] AS/NZS 4360:2004, *Risk management*, AS/NZS.
- [23] R.R. Moeller, *COSO Enterprise Risk Management*, Wiley, New Jersey, 2007.
- [24] COSO, *Internal control – integrated framework*. www.coso.org (1992)
- [25] Public Company Accounting Oversight Board, *Auditing Standard No. 5 – An Audit of Internal Control Over Financial Reporting That Is Integrated with An Audit of Financial Statements*. http://www.pcaobus.org/Rules/Rules_of_the_Board/Auditing_Standard_5.pdf (2007)
- [26] P.P. Gupta, *Internal Control. COSO 1992 Control Framework and Management Reporting on Internal Control: Survey and Analysis of Implementation Practices*. <http://ssrn.com/abstract=1417604> (2009)
- [27] M. Rath, R. Sponholz, *IT-Compliance: Erfolgreiches Management regulatorischer Anforderungen*, Schmidt, Berlin, 2009.
- [28] W.W. Deming, *Out of the Crisis*, MIT, Cambridge (MA), 1982.
- [29] G. Tennant, *Six Sigma: SPC and TQM in Manufacturing and Services*, Gower, Aldershot, 2001.
- [30] N. Racz, E. Weippl, A. Seufert, A process model for integrated IT governance, risk, and compliance management, J. Barzdins, M. Kirikova (eds.), *Databases and Information Systems*, Proceedings of the Ninth International Baltic Conference, Baltic DB&IS (2010), 155–170.
- [31] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, *MIS Quarterly* **28:1** (2004), 75–105.

Towards Model Transformation between SecureUML and UMLsec for Role-based Access Control

Raimundas MATULEVIČIUS^{a, b, 1} and Marlon DUMAS^{a, b}

^a*Institute of Computer Science, University of Tartu,
J. Liivi 2, 50409 Tartu, Estonia*

^b*Software Technology and Application Competence Center,
Ülikooli 4, 51003 Tartu, Estonia*

Abstract. Nowadays security has become an important aspect in information systems engineering. A mainstream method for information system security is Role-based Access Control (RBAC), which restricts system access to authorised users. Recently different authors have proposed a number of modelling languages (e.g., abuse cases, misuse cases, secure *i**, secure Tropos, and KAOS extensions to security) that facilitate the documentation and analysis of security aspects. However it is unclear if these languages support the full spectrum of RBAC specification needs. In this paper we selected two security modelling languages, namely SecureUML and UMLsec. Based on the literature study and on the running example we systematically investigate how these languages could be used for RBAC. Our observations indicate that, although both approaches originate from the *de-facto* industry standard UML, they are not competitors. Rather they complement each other: SecureUML helps defining static RBAC aspects; UMLsec is recommended for dynamic RBAC analysis. Thus, the combined use of both approaches would provide a more comprehensive approach to secure information system development. As a step towards enabling the combined use of SecureUML and UMLSec, this paper outlines a mapping transformation between these two languages.

Keywords. Model-driven security, SecureUML, UMLsec, role-based access control, security modelling languages

Introduction

Nowadays information systems play an important role in everybody's life. They are used in different areas and domains, including banking, education, medicine and others. People need to deal with information, which at many cases is confidential and should not be accessible for un-authorised use. Thus, ensuring security of information systems is a necessity rather than an option. Security is usually defined along four dimensions [1]: integrity (ensuring information is not altered), non-repudiation (ensuring receiving parties cannot renege on the receipt of information), authentication (confirming the originator and intended recipient of information) and confidentiality (ensuring information is shared only among authorised parties). In this paper, we focus on the

¹ Corresponding Author.

latter dimension and specifically, on one mechanism for ensuring confidentiality, namely Role-Based Access Control (RBAC) that restricts information access to authorised users.

Although security is an important aspect in information systems engineering, the literature [2], [3] reports that security concerns are often raised only when the system is about to be deployed or is already in use, or in the best-case security is considered only during the late system development stages (e.g., implementation). This is a serious hindrance to secure system development, since the early stages (e.g., requirements and design) are the place where system security concerns should be discovered and security trade-offs should be analysed. One possible way to guide such an analysis is suggested by the model-driven security approaches. For instance, Abuse frames [4] suggest means to consider security during the early phases of requirements engineering. Secure *i** [5] addresses security trade-offs. KAOS [6] was augmented with anti-goal models designed to elicit attackers' rationales. In [7] Tropos has been extended with the notions of ownership, permission and trust. Another version of Secure Tropos suggested in [8] models security using security constraints and attack methods. Abuse cases [9], misuse cases [3] and mal-activity diagrams [10] address security concerns through negative scenarios executed by the attacker.

All these modelling approaches could be applied to model RBAC in a system [11], however they are rather general than specific. In the literature we have observed that there are two modelling approaches – SecureUML [12][13] and UMLsec [2][14] – that, actually, contain targeted concepts for RBAC. In [15] we report on a comparison of SecureUML and UMLsec according to the literature study and some illustrative example [16]. Our observations are that these approaches are not competitors when it comes to RBAC modelling, but they rather complement each other by addressing different security modelling perspectives. Therefore, an opportunity exists to employ these approaches in combination, so as to obtain more comprehensive information systems security models, particularly with respect to RBAC. As a first step towards enabling a combined use of SecureUML and UMLSec, we outline a set of rules for transforming a SecureUML model to an UMLsec model and vice versa. In this way, developers can start with a model in either of these approaches and later on, enrich this initial model using the other approach.

The structure of the paper is as follows: in Section 1 we introduce the general RBAC model and two modelling approaches – SecureUML and UMLsec. In Section 2 we compare SecureUML and UMLsec using the illustrative example. Based on the comparison results we define a set of semi-automatic transformation rules in Section 3. Finally, in Section 4 we conclude our study and present some future work.

1. Background

In this section we present the major artefacts discussed in this paper. Firstly, we recall the general RBAC model [17]. Next, we present the major principles of SecureUML and UMLsec.

1.1. Role-based Access Control

The main elements of the RBAC model [17] are *Users*, *Roles*, *Objects*, *Operations*, and *Permissions*. A *User* is typically defined as a human being or a software agent. A *Role*

is a job function within the context of an organisation. Role refers to authority and responsibility conferred on the user assigned to this role. *Permissions* are approvals to perform one or more *Operations* on one or more protected *Objects*. An *Operation* is an executable sequence of actions that can be initiated by the system entities. An *Object* is a protected system resource (or a set of resources). Two major relationships in this model are *User assignment* and *Permission assignment*. *User assignment* relationship describes how users are assigned to their roles. *Permission assignment* relationship characterises the set of privileges assigned to a *Role*.

1.2. SecureUML

The SecureUML meta-model [12] [13] is based on the RBAC model. It defines the abstract syntax (see Figure 1) to annotate UML diagrams with information pertaining to access control. The meta-model introduces concepts like User, Role, and Permission as well as relationships between them. Protected resources are expressed using the standard UML elements (concept of ModelElement). In addition ResourceSet represents a user defined set of model elements used to define permissions and authorisation constraints.

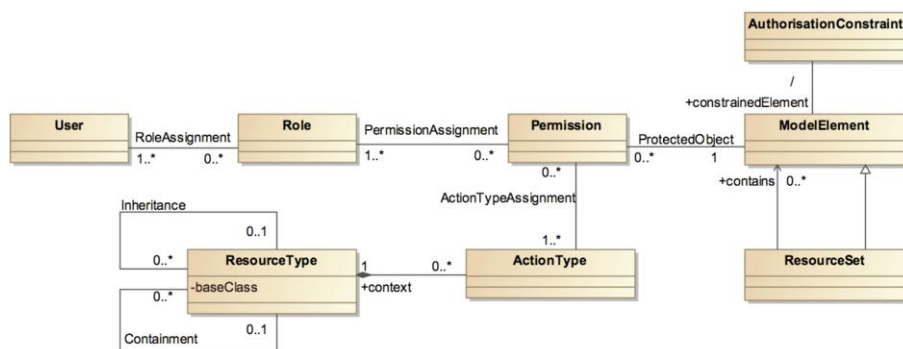


Figure 1. SecureUML meta-model (adapted from [12], [13])

The semantics of Permission is defined through ActionType elements used to classify permissions. Here every ActionType represents a class of security-relevant operations (e.g., *read*, *change*, *delete*, and etc) on a particular type of protected resource. On another hand a ResourceType defines all action types available for a particular meta-model type. An AuthorisationConstraint is a part of the access control policy. It expresses a precondition imposed to every call to an operation of a particular resource. This precondition usually depends on the dynamic state of the resource, the current call, or the environment. The authorisation constraint is attached either directly or indirectly, via permissions, to a particular model element representing a protected resource. The concrete syntax of SecureUML is illustrated in Figure 2 and discussed in Section 2.1.

1.3. UMLsec

A major purpose of security modelling is to define mechanisms to satisfy security criteria, such as confidentiality and integrity [18]. To support this activity UMLSec [2]

[14] is defined as a UML profile extension using stereotypes, tagged values and constraints (see Table 1). Constraints specify security requirements. Threat specifications correspond to actions taken by the adversary. Thus, different threat scenarios can be specified based on adversary strengths.

Table 1. UMLsec stereotypes (adapted from [2] [14]). The scope of this paper is highlighted in *italic*

Stereotypes	Base class	Tags	Constraints	Description
fair exchange	subsystem	start, stop, adversary	after start eventually reach stop	enforce fair exchange
<i>Rbac</i>	<i>subsystem</i>	<i>protected, role, right</i>	<i>only permitted activities executed</i>	<i>enforces RBAC</i>
Internet encrypted	link			Internet connection is encrypted
smart card	node			smart card node
critical	subsystem, object	secrecy, integrity, authenticity, high, fresh		critical object
data security	subsystem	adversary, integrity, authenticity	provides secrecy, integrity, authenticity, freshness	basic data security constraints
guarded access	subsystem		guarded object accessed through guards	access control using guarded objects
guarded	object	Guard		guarded object

A subset of UMLsec that is directly relevant to this study is the role-based access control stereotype – <<rbac>> – its tagged values and constraints [2]. This stereotype enforces RBAC in the business process specified in the activity diagram. It has three associated tags {protected}, {role}, and {right}. The tag {protected} describes the states in the activity diagram, the access to whose activities should be protected. The {role} tag may have as its value a list of pairs (*actor, role*) where *actor* is an actor in the activity diagram, and *role* is a role. The tag {right} has as its value a list of pairs (*role, right*) where *role* is a role and *right* represents the right to access a protected resource. The associated constraint requires that the actors in the activity diagram only perform actions for which they have the appropriate rights. The application of the <<rbac>> stereotype is illustrated in Section 2.2.

2. Comparison

In [15] we compare SecureUML and UMLsec based on the literature study. Like in [11], there we notice the limitation of SecureUML to indicate security criteria. We also observe that the UMLsec application follows the standard security modelling methods [18]. In contradiction to [19] we observe that UMLsec provides means for RBAC modelling. Thus, we suggest that both approaches can complement each other and result in more complete specifications of secure information systems.

In this work we compare modelling constructs of SecureUML and UMLsec following the *Meeting scheduler* example [16], which is described as follows: *Meeting initiator* needs to organise a *top-secret* meeting. He needs to invite potential *Meeting participants* and find a suitable meeting *place* and *time*. In order to ease his task *Meeting initiator* decides to use a *Meeting scheduler system* for sending invitations,

merging availability dates and informing the *Meeting participants*. Since the *Meeting* is top secret, the *Meeting scheduler system* must apply appropriate security policy for the *Meeting agreement* (place and time). This means, the *time* and *place* could be entered and changed only by the *Meeting initiator* and could be viewed only by the invited *Meeting participants*. In other words, no unintended audience should get access to the *Meeting agreement*. We will illustrate how this problem can be modelled with SecureUML and with UMLsec.

2.1. SecureUML Model

In Figure 2 we present a SecureUML model to illustrate RBAC policy for the *Meeting Scheduler System*. Here we define three users *Bob*, *Ann* and *John*, who play different roles in the system. We also present that a resource (*MeetingAgreement*), which characterise *place* and *time* of the meeting, needs to be secured. Thus, a certain restriction on changing the state (changing the value of the attributes *place* and *time*) of this resource needs to be defined for the role *MeetingInitiator* and role *MeetingParticipant*.

Association class *InitiatorPermissions* characterises two *actions* allowed for the *MeetingInitiator*: (i) action *enterAgreementDetails* (of type Insert) defines that *MeetingInitiator* can enter *time* and *date* by executing operation *setTimePlace()* (see class *MeetingAgreement*), and (ii) action *changeMeetingInfo* (of type Update) allows changing *place* and *time* of the *MeetingAgreement* by executing operation *changeTimePlace()* (see class *MeetingAgreement*). To strengthen these permissions we define *authorisation constraints* AC#1 and AC#2:

```
context MeetingAgreement::setTimePlace():void                                     AC#1
pre: self.roleInitiator.assignedUser ->
    exists(i | i.assignedUser = "Bob")
```

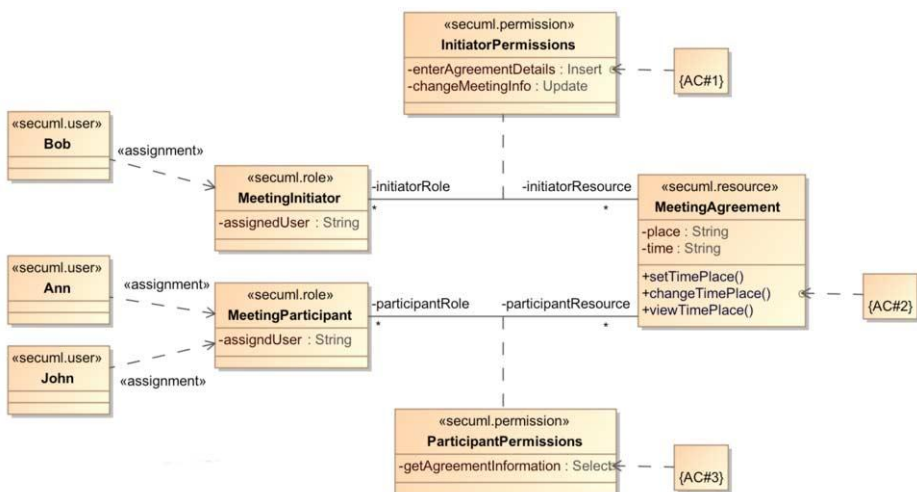


Figure 2. Meeting Scheduler with SecureUML

Authorisation constraint AC#1 means that operation *setTimePlace()* (of class *MeetingAgreement*) can be executed (enter *time* and *place*), by one user *Bob* assigned

to a role *MeetingInitiator*². Similarly, the authorisation constraint AC#2 defines restriction for operation *changeTimePlace()* (of class *MeetingAgreement*):

```
context MeetingAgreement::changeTimePlace():void          AC#2
pre: self.roleInitiator.assignedUser ->
    exists(i|i.assignedUser = "Bob")
```

Association class *ParticipantPermissions* defines a restriction for the *MeetingParticipant* role. It defines an action *getAgreementInformation* (of type Select) that says that only *MeetingParticipant* can view *place* and *time* defined in the *MeetingAgreement*. To enforce this permission an authorisation constraint AC#3 is defined:

```
context MeetingAgreement::viewTimePlace():void          AC#3
pre: self.roleParticipant->
    exists (p1|p1.assignedUser="Ann")and
    self.roleParticipant->
    exists (p2|p2.assignedUser="John")and
    self.roleParticipant->size = 2
```

Authorisation constraint AC#3 says that only users *Ann* and *John* who have an assigned role *MeetingParticipant* can execute an operation *viewTimePlace()* (of class *MeetingAgreement*).

2.2. UMLsec Model

Figure 3 illustrates application of UMLsec to model the *Meeting Scheduler System*. Here we define an activity diagram, which describes an interaction between *MeetingInitiator*, *MeetingAgreement*, and *MeetingParticipant*. The diagram specifies that *MeetingInitiator* can insert meeting time and date. Next *MeetingParticipant* is able to check if the time and place are suitable to him. If the agreement is not OK, *MeetingParticipant* requests *MeetingInitiator* to update. After the agreement (*time* and *date* of the meeting) data are updated *MeetingParticipant* can check them again for suitability.

This diagram carries an <<rbac>> stereotype, meaning that the security policy needs to be applied to the protected actions. For instance, the *MeetingInitiator's* action Insert meeting time and date leads to the action Set time and date for the *MeetingAgreement*. Set time and date is executed if and only if there exists an associated tag, that defines the following: (i) Set time and date is a protected action, (ii) *Bob* plays a role of *MeetingInitiator*, and (iii) *MeetingInitiator* enforces the action Set time and date. In the activity diagram this associated tag (AT#1) is defined as follows:

```
{protected = Set time and date}          AT#1
{role = (Bob, MeetingInitiator)}
{right = (MeetingInitiator, Set time and place)}
```

² As illustrated in [12] SecureUML model might contain both objects (e.g., Bob, Ann, and John) and classes (e.g., MeetingInitiator, MeetingParticipant). This results in restrictive authorisation constraints AC#1, AC#2 and AC#3. In case the user assignment relationship was not specified, the precondition might be expressed like `self.roleInitiator.assignedUser=caller`, where `caller` is a set of users (assigned to a role) on behalf of whom the operation is executed.

Similarly, the sets of associated tags are defined for other two protected actions *View time and date* (AT#2) and *Change time and date* (AT#2). Note that both *Ann* and *John* can initiate execution of action *View time and date* (AT#2), since they both play the role of *MeetingParticipant*.

```
{protected = View time and date}                                     AT#2
{role = ([Ann, John], MeetingParticipant)}
{right = (MeetingParticipant, View time and place)}
```

```
{protected = Change time and date}                                   AT#3
{role = (Bob, MeetingInitiator)}
{right = (MeetingInitiator, Change time and place)}
```

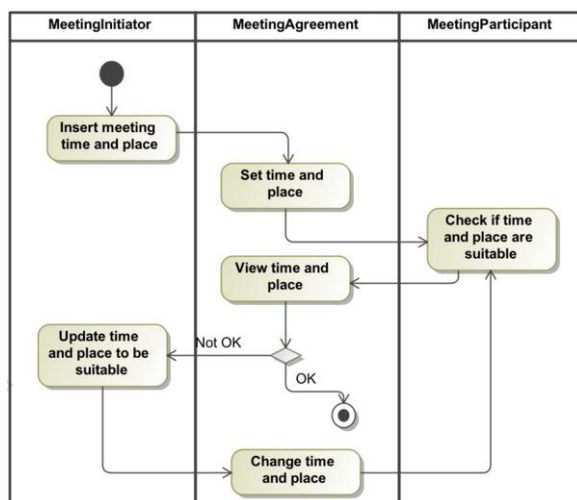


Figure 3. Meeting Scheduler with UMLsec

2.3. Comparing SecureUML and UMLsec Constructs

In Table 2 we compare the RBAC modelling using SecureUML and UMLsec. We base our comparison on the RBAC model [17] presented in Section 2.1. In this comparison we review which constructs are used for expressing the RBAC concepts and relationships. In Table 2 we also provide construct examples from Figure 2 and Figure 3.

Firstly, we observe that both approaches cover all RBAC concepts and relationships. This means that both approaches can express security policies through RBAC. Secondly, SecureUML addresses RBAC through the defined stereotypes, while than UMLsec expresses RBAC concepts and relationships through the associated tags and their values. At the example level, both approaches use the same (e.g., for *Users*, *Roles*, and *Objects*) or very similar (e.g., for *Operations*) labels for the RBAC concepts.

Some modelling and labelling differences are observed when it comes to relationship definition. In SecureUML *User assignment* relationship is modelled through a UML-stereotyped dependency without defining any label. In UMLsec the specific associated tag – {role} – is defined for a user-to-role assignment. Different labelling is also used to specify *Permission assignments*. In SecureUML this is done

through stereotyped association classes, which might carry a name depending on the modelled context. In UMLsec the associated tag – {right} – is used for this purpose.

Table 2. Comparison of RBAC modelling using SecureUML and UMLsec

	RBAC concepts	SecureUML		UMLsec	
		Construct	Example	Construct	Example
1	Users (concept)	Class stereotype <<secuml.user>>	Bob, Ann, and John	<i>Actor</i> value of the associated tag {role}	“Bob”, “Ann”, and “John”
2	User assignment (relationship)	Dependency stereotype <<assignment>>	Dependency between classes such as Bob and MeetingInitiator, and Ann or John and MeetingParticipant	Associated tag {role}	{role = (Bob, MeetingInitiator)} {role = ([Ann, John], MeetingParticipant)}
3	Roles (concept)	Class stereotype <<secuml.role>>	MeetingInitiator and MeetingParticipant	<i>Activity partition</i> <i>Role</i> value of the associated tag {role}	MeetingInitiator and MeetingParticipant “MeetingInitiator” and “MeetingParticipant”
4	Permission assignment (relationship)	Association class stereotype <<secuml.permission>>	Operations of association classes InitiatorPermissions and ParticipantPermissions	<i>Action</i> Associated tag {right}	Insert meeting time and place, Check if time and place are suitable, and Update time and place to be suitable {right = (MeetingInitiator, Set time and date)} {right = (MeetingParticipant, View time and date)} {right = (MeetingInitiator, Change time and date)}
5	Objects (concept)	Class stereotype <<secuml.resource>>	MeetingAgreement	<i>Activity partition</i>	MeetingAgreement
6	Operations (concept)	Operations of <<secuml.resource>> class	setTimePlace(), changeTimePlace() and viewTimePlace()	<i>Action</i> Associated tag {protected}	Set time and place, View time and place, and Change time and place {protected = (Set time and place)}, {protected = (View time and place)}, {protected = (Change time and place)}
7	Permissions (concept)	Authorisation constraint	AC#1, AC#2, and AC#3	Not defined	Not defined

Finally, we note, that UMLsec does not provide explicit means to define *Permission* itself. This is rather left implicitly at the diagram level when defining the values for all the associated tags ({protected}, {role}, and {right}). In SecureUML *Permissions* are explicitly defined through authorisation constraints expressed in OCL (see, for instance, the preconditions in AC#1, AC#2, and AC#3).

3. Transformation Rules

Following our comparative analysis of SecureUML and UMLsec for the RBAC we have developed a set of rules for transforming SecureUML models to UMLsec models

and vice versa. Some of these rules could be directly implemented in the modelling tools, thus resulting in the semi-automated support for RBAC model transformation. In order to illustrate our proposal we will incrementally show how the transformation rules would be applied to the SecureUML class diagram (e.g., *Meeting Scheduler* example in see Figure 2) to translate it to the UMLsec activity diagram (given in Figure 4).

3.1. Model Transformation from SecureUML to UMLsec

Below we define four transformation rules to transform a model from SecureUML to UMLsec:

SU1. A class with a stereotype `<<secuml.resource>>` is transformed to an activity partition in the UMLsec model (Table 2, line 5), and the operations of this class become actions belonging to this partition (Table 3, line 6). In addition, each operation becomes a value the UMLsec associated tag `{protected}`.

Example: the class *MeetingAgreement* (see Figure 2) is represented as activity partition in Figure 4. The operations *setTimePlace()*, *changeTimePlace()*, and *viewTimePlace()* are shown as actions in this partition. Three associated tags (see Table 2) `{protected}` are defined: `{protected=(setTimePlace)}`, `{protected=(changeTimePlace)}`, and `{protected=(viewTimePlace)}`;

SU2. A relationship with a stereotype `<<assignment>>` relationship used to connect users and their roles is transformed to an associated tag `{role}`.

Example: From Figure 2 we specify associated tags `{role=(Bob, MeetingScheduler)}`, `{role=(Ann, MeetingParticipant)}`, and `{role=(John, MeetingParticipant)}`, as provided in Table 2.

SU3. A SecureUML class with the stereotype `<<secuml.roles>>` is transformed to the UMLsec activity partition (Table 2, line 3). The attributes of an association class that connects the `<<secuml.roles>>` class with `<<secuml.resource>>` class, become actions in the corresponding activity partition (Table 2, line 4).

Example: in Figure 4 we define activity partitions *MeetingInitiator* (with the actions *enterAgreementDetails* and *changeMeetingInfo* grabbed from the class *InitiatorPermission*) and *MeetingParticipant* (with the action *getAgreementInformation* defined from the class *ParticipantPermissions*);

SU4. The SecureUML association class with the stereotype `<<secuml.permission>>` defines the *role* value for the UMLsec associated tag `{right}` (Table 2, line 4). The value of *right* can be determined from the authorisation constraint defined for the attribute of the SecureUML association class.

Example: this rules leads to the association tags `{right =(MeetingInitiator, setTimePlace)}` and `{right =(MeetingParticipant, viewTimePlace)}` captured in Table 3.

Note: The authorisation constraint might help to identify the relationship between two actions as shown in Figure 4 between actions *enterAgreementDetails* and *setTimePlace*, and between actions *getAgreementInformation* and *viewTimePlace*.

Note: Complete definition of the associated tag `{right}` is not always possible because (i) it might be that no association constraint is defined at all (no additional security enforcement is needed), or (ii) an authorisation constraint might be

defined at a different place (e.g., to strengthen the operations of the <<secuml.resource>> classes) in the SecureUML model as shown for the authorisation constraint AC#2 in Figure 3.

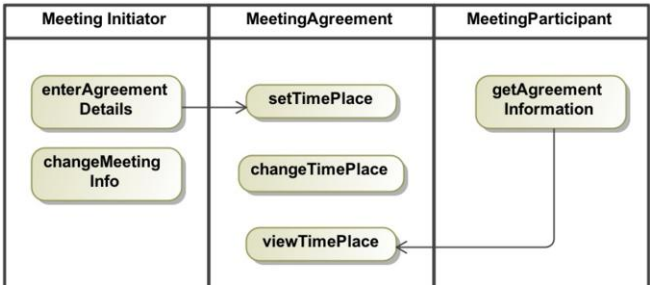


Figure 4. Transformed UMLsec model
(only automatic translations according to rules SU1-SU4 are shown)

In Table 3 we summarise the resulting association tags. However, as mentioned above, we are not able to determine all the association tags required in the model. For example the {right} association tag is captured from the association class <<secuml.permission>> and its link to the authorisation constraint. However not all association constraints are links to the association classes <<secuml.permission>>. For example, in Figure 2 the authorisation constraint AC#2 is linked to class *MeetingAgreement*. This means that the association tag {right =(MeetingInitiator, changeTimePlace)} needs to be written manually by the developer. This means that manually two actions need to be linked by a control flow as well (e.g., from *ChangeMeetingInfo* to *changeTimePlace*), given that SecureUML is not concerned with behavioural aspects.

Table 3. Automatically translated association tags

Transformation rule	Association tags
SU1.	{protected=(setTimePlace)} {protected=(changeTimePlace)} {protected=(viewTimePlace)}
SU2.	{role=(Bob, MeetingScheduler)} {role=(Ann, MeetingParticipant)} {role=(John, MeetingParticipant)}
SU4.	{right =(MeetingInitiator, setTimePlace)} {right =(MeetingParticipant, viewTimePlace)}

To complete the UMLsec activity diagram (e.g., Figure 4) a developer needs to specify information that was not possible to capture from the SecureUML diagram. For instance the developer needs to define initial node (e.g., to *enterAgreementDetails* action) and activity final node (e.g., from *viewTimePlace* action). Other control flows (including the conditionals ones) needs also to be specified. For instance in Figure 4 control flows between *setTimePlace* and *getAgreementInformation*, *viewTimePlace* and *changeMeetingInfo*, and *changeTimePlace* and *getAgreementInformation* might define a logical sequence of activity that corresponds to the one in Figure 3.

3.2. Model Transformation from UMLsec to SecureUML

The transformation from UMLsec to SecureUML is defined by means of four rules. To illustrate these rules, we will analyse the *Meeting Scheduler* example presented in Figure 3. The resulting SecureUML class diagram is given in Figure 5.

US1. Association tags {protected} allow us identify the operations that belong to secured resource (Table 2, line 6). We transform the activity partitions, which hold these operations to the SecureUML class with a stereotype <<secuml.resource>> (Table 2, line 5).

Example: in Figure 5 we define a class *MeetingAgreement* which has three operations *Set time and place()*, *View time and place()*, and *Change time and place()*.

US2. In the UMLsec model the activity partitions that do not hold secured protected actions, can be transformed to the <<secuml.role>> classes (Table 3, line 3).

Example: In Figure 5 we define two <<secuml.role>> classes: *MeetingInitiator* and *MeetingParticipant*.

US3. Association tag {roles} allows us to identify the <<assignment>> dependency relationship (Table 2, line 2) between classes of users defined with a stereotype <<secuml.user>>, and their roles presented with a stereotype <<secuml.role>>.

Example: in Figure 5 we define three <<assignment>> dependency links: (i) between *Bob* and *MeetingInitiator*, (ii) between *Ann* and *MeetingParticipant*, and (iii) between *John* and *MeetingParticipant*.

US4. From UMLsec association tag {right} we are able to identify on which operations a role can perform security actions (Table 2, line 4). Thus, from each occurrence of this association tag in the SecureUML model, a corresponding association class between a <<umlsec.role>> and a <<umlsec.resource>> is introduced.

Example: In Figure 5 we define two association classes: (i) between *MeetingInitiator* and *MeetingAgreement*, and (ii) between *MeetingParticipant* and *MeetingAgreement*.

Furthermore, from the UMLsec activity diagram (Table 2, line 4) we are able to identify what exact security actions are carried towards the secured operations: these are unprotected actions performed before the protected ones.

Example: in Figure 3 the action *Insert meeting time and place* is performed before protected action *Set time and place*. We transform the *Insert meeting time and place* action to the attribute of the association class between *MeetingInitiator* and *MeetingAgreement* (see Figure 5). The similar transformations are performed for the attribute *Update time and place to be suitable* for the same association class, and for the attribute *Check if time and place are suitable* for the association class between *MeetingParticipant* and *MeetingAgreement*.

Note: we are not able to identify the type of security actions.

The SecureUML model needs to be completed manually with the information, which is not captured from the UMLsec model. Specifically, the developer needs to introduce the following information:

- the attributes of the <<secuml.resource>> class that define the state of the secured resource(s). For example, in Figure 5 the class *MeetingAgreement* should be complemented with attributes *place:String* and *time:String*.

- all the necessary authorisation constraints for the SecureUML model. For instance for Figure 5 to correspond Figure 2 the authorisation constraints AC#1, AC#2, and AC#3 have to be defined;
- multiplicities for all the association relationships. For example, multiplicities for associations (see Figure 5) between *MeetingInitiator* and *MeetingAgreement*, *MeetingParticipant* and *MeetingAgreement* have to be defined;
- names for the association classes. For instance in Figure 5 names for classes with the <<secuml.permission>> stereotype have to be specified;
- action types for the identified actions. For example, for action *Insert meeting time and place* action type is Insert, for *Update time and place to be suitable* action type is Update, and for *Check if time and place are suitable* action type is Select.

Finally, the developer should also include additional model attributes as required. For instance in order the translated model (Figure 5) would correspond to the *Meeting Scheduler* in Figure 2, one needs to define attributes *assignedUser:String* for classes *MeetingInitiator* and *MeetingParticipant*. These attributes would carry the information about the users assigned for these roles.

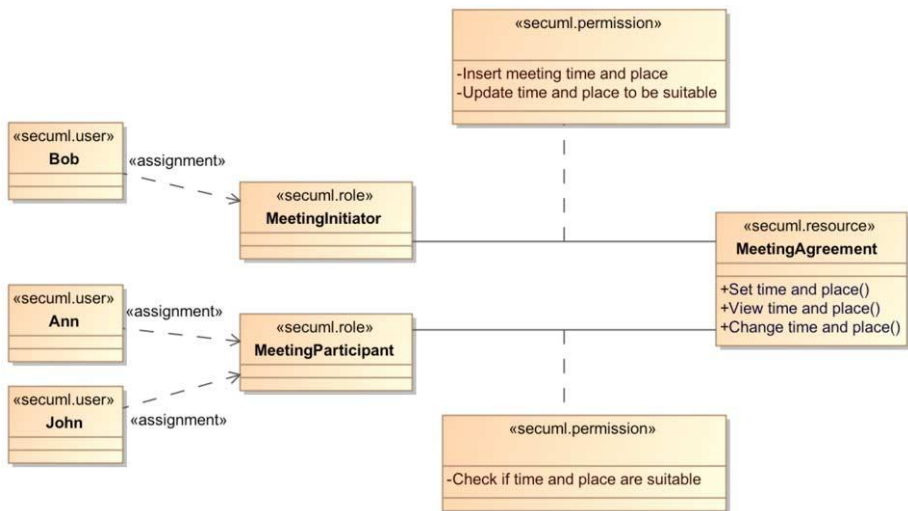


Figure 5. Transformed SecureUML model
(only automatic translations according to rules US1-US4 are shown)

3.3. Threats to Study Validity

This study is not without limitations. Firstly, we need to note that our analysis is of limited scope, as it is based on the literature (e.g., [12], [13] for SecureUML and [2] [14] for UMLsec) and on a simple example (e.g., *Meeting Scheduler System* [16]). It might be the case that if we carried out an extensive empirical study or a set of benchmarking examples we would receive different comparison results and different rules for transformation. Secondly, although we followed the theory as close as possible, our modelling example (in Figure 2 and Figure 3) carries a certain degree of subjectivity regarding the modelling decisions.

Thirdly, we should note that for our analysis we have selected only extracts of the modelling approaches. This is especially true for the UMLsec, where we focussed only

on the means to define RBAC aspects. In addition to this UMLsec also provides other stereotypes (e.g., <<guarded access>> and <<guarded>>) that are used to deal with access control policies (without considering role assignments).

In our comparative example we skipped the application of UMLsec throughout the security risk management process [18]. For example, in Figure 3 we could consider MeetingAgreement actions as *assets* and analyse the security criteria, such as *confidentiality*, *integrity*, and *availability* of the meeting agreement. Then, we could define a new activity partition, which would illustrate how an *attacker* could exploit system *vulnerabilities* in order to break the security. However, in this paper we specifically focus on the definition of the security solution (through RBAC), because the goal is to contrast UMLSec with SecureUML, and the latter does not support security risk management.

4. Conclusions and Future Work

In this paper we have analysed how SecureUML and UMLsec can help defining security policies through the role-based access control mechanism. We observe that both SecureUML and UMLsec are applicable to model RBAC solutions. Table 2 illustrates that both approaches have means to address the RBAC concepts and relationships. The strong feature of SecureUML is the explicit definition of Permissions through authorisation constraints using OCL. On another hand our general comparison showed that at the methodological level, SecureUML only focuses on the solution domain. Meanwhile, UMLsec provides means to identify and to consider system risks, to determine system vulnerabilities, and also to develop solutions to mitigate the identified risks (RBAC being one of such solutions) [15].

Although both approaches originate from UML, SecureUML and UMLsec focus on different modelling perspectives to define security policies. SecureUML is used to model static characteristics of RBAC, thus, it is applied on top of class diagrams. UMLsec is used to model dynamic characteristics of RBAC³, thus it relies heavily on activity diagrams. Taking into account that system modelling needs to be addressed from different modelling perspectives, this means that both approaches are not competitors, but they rather complement each other by providing different viewpoints to the secure system.

In this work we outline a set of transformation rules in order to perform a semi-automatic translation from SecureUML to UMLsec and vice-versa. These rules can be implemented in a CASE tool in order to support a multi-perspective approach to RBAC definition. The implementation and empirical testing of the transformation rules currently remains a future work. Another direction for future work is to define an incremental transformation that would allow developers to maintain pairs of concurrently evolving SecureUML and UMLSec models synchronised.

³ This is relevant only for the UMLsec <<rbac>> stereotype, other UMLsec stereotypes can be applied at different model types.

Acknowledgment

This research is funded by Logica and the European Regional Development Funds through the Estonian Competence Centre Programme.

References

- [1] W. Caelli, D. Longley, M. Shain, *Information Security Handbook*, Macmillan Publishers, 1991.
- [2] J. Jurjens, *Secure Systems Development with UML*, Springer-Verlag Berlin Heidelberg, 2005.
- [3] G. Sindre, A.L. Opdahl, Eliciting Security Requirements with Misuse Cases, *Requirements Engineering Journal* **10** (1) (2005), 34–44
- [4] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, Using Abuse Frames to Bound the Scope of Security Problems, *Proceedings of the 12th IEEE international Conference on Requirements Engineering*, IEEE Computer Society, 2004, 354–355.
- [5] G. Elahi, E. Yu, A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs, In Parent *et al.* (eds.), *Proceedings of the 26th International Conference on Conceptual Modelling*, 2007.
- [6] A. van Lamsweerde, Elaborating Security Requirements by Construction of Intentional Anti-models, *Proceedings of the 26th International Conference on Software Engineering*, IEEE Computer Society, 2004, 148–157
- [7] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Modeling Security Requirements Through Ownership, Permission and Delegation. *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, IEEE Computer Society, 2005.
- [8] H. Mouratidis, Analysing Security Requirements of Information Systems using Tropos, *Proceedings 1st Annual Conference on Advances in Computing and Technology*, 2006, 55 – 64.
- [9] J. McDermott, C. Fox, Using Abuse Case Models for Security Requirements Analysis. *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999.
- [10] G. Sindre, Mal-activity Diagrams for Capturing Attacks on Business Processes, *Proceedings of the Working Conference on Requirements Engineering: Foundation for Software Quality*, Springer-Verlag Berlin Heidelberg, 2007, 355–366.
- [11] A. Bandara, H. Shinpei, J. Jurjens, H. Kaiya, A. Kubo, R. Laney, H. Mouratidis, A. Nhlabatsi, B. Nuseibeh, Y. Tahara, T. Tun, H. Washizaki, N. Yoshioka, Y. Yu, Security Patterns: Comparing Modelling Approaches, *Technical Report No 1009/06*, Department of Computing Faculty of mathematics, Computing Technology, The Open University, 2009.
- [12] D. Basin, J. Doser, T. Lodderstedt, Model Driven Security: from UML Models to Access Control Infrastructure, *ACM Transactions on Software Engineering and Methodology*, **15** (1), (2006) 39–91.
- [13] T. Lodderstedt, D. Basin, J. Doser, SecureUML: A UML-based Modeling Language for Model-driven Security. *Proceedings of the 5th International Conference on The Unified Modeling Language*, Springer-Verlag, 2002, 426–441.
- [14] J. Jurjens, UMLsec: Extending UML for Secure Systems Development. *Proceedings of the 5th International Conference on The Unified Modeling Language*, Springer-Verlag, 2002, 412–425.
- [15] R. Matulevičius, M. Dumas, A Comparison of SecureUML and UMLsec for Role-based Access Control, *Proceedings of the 9th Conference on Databases and Information Systems*, 2010, 171–185
- [16] M. S. Feather, S. Fickas, A. Finkelstein, A. van Lamsweerde, Requirements and Specification Exemplars, *Automated Software Engineering*, **4** (1997) 419–438.
- [17] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, Proposed NIST Standard for Role-based Access Control, *ACM Transactions on Information and System Security*, **4**(3) (2001), 224–274.
- [18] N. Mayer N., *Model-based Management of Information System Security Risk*, PhD Thesis, University of Namur, 2009
- [19] K. R. Jayaram, A. P. Mathur, Software Engineering for Secure Software – State of the Art: a Survey. *Technical report CERIAS TR 2005-67*, Department of Computer Sciences & CERIAS, Purdue University (2005)

Information Security Management Method for Households

Ilze MURANE¹

University of Latvia, Riga, Latvia

Abstract. Threats to the home computer network are the same as for business organizations. Business has spent years to improve information security management; there are risk and security management professionals. Households usually lack knowledge and resources to protect electronic information, but security is necessary for everybody. The article deals with threats stemming from new technological possibilities that are entering households. It outlines the importance of information security management and security awareness in households. It also describes a solution for encouraging security awareness in relation to family electronic information by defining roles and responsibilities in the information security management process. The paper suggests a method for assessing information security risks in a private environment.

Keywords. Awareness, household, information security, responsibility

Introduction

Access to and the quality of Internet services in Latvia are among the best in the world [1], and this inevitably promotes interest in those of our computers which are not protected too well. According to the Central Statistical Board, 58% of households in Latvia have had Internet access at the beginning of 2009, and 86.6% of those connections have been broadband links [2].

The use of technologies at home is also developing. Often we must talk not just about protecting the computer, but also about protecting the much broader in-home computer network and its structure. For home-based computer networks, there are important issues such as network policy, authentication, authorization, etc. There are studies which have offered solutions in this area [3].

Tens of thousands of Internet users found their access blocked for many hours in early January this year [4]. The service provider Vispa tracked the route of the attack, and it turned out that some “zombified” Latvian computers or botnets are to blame for this. They were being used by “owners” who were complete strangers. It is very unlikely that the guilty parties are to be found. Attacks of this type can be based on nothing more than hooliganism, but they can also involve dishonest competitors or vengeful staff members who have been sacked. True attackers can come from any place in the world. They can be neighbors of the victim. Botnets are created in unprotected or poorly protected computers by sending in special software that goes to work without

¹ University of Latvia, Raina blvd. 19, Riga, Latvia; E-mail: ilze.murane@lu.lv.

the computer's owner even knowing about it. Most of the owners of computers that have been used for attacks probably don't even know that this has happened.

As is the case in most of the world, people of Latvia have also developed experience and knowledge about computer security more slowly than technologies and related threats are progressing.

Technological protection is not enough, it turns out. There are many different tools of social engineering that are used for the purpose of getting people to disclose different kinds of information. Social engineering is defined in different ways, but one definition is that it is an artful way of getting others to do what you want them to do. This "art" is ancient, but in recent times it has been used quite widely in the field of information technologies, because people who don't know about security issues or are "scared" in an unsafe environment are easier to fool. Accordingly, broader solutions are needed. Information security is a subject that is increasingly being discussed also outside the fields of computer science and mathematics. Professionals from different areas are working together in pursuit of solutions. In particular, economists and psychologists are cited as participants in this process [5].

For children and adolescents, who are among the most active users of online resources, threats can occur more frequently. Social network sites have "developed significant cultural resonance amongst American teens in a short period of time" [6]. Education for young people is very important so that users can better understand the potential threats which occur when they publish too much information about themselves [7].

Management of information security is gradually becoming a subject that is also considered by top managers at companies. This makes it possible to take the necessary organizational decisions and to ensure the resources that are needed for that purpose. Households use the same Internet and face the same threats, however. Perhaps it is the case that the best aspects of information security tools that are used at the corporate level could also be used at households.

Chapter 1 of the paper discusses the historic development of the Internet. Chapter 2 describes threats and information security management in households. Chapter 3 emphasizes the necessity of security governance and awareness, while Chapter 4 sets out sample security rules for households. Chapter 5 describes a method for assessing information security in a private environment. The paper also poses questions that are relevant for anybody working with his/her information in an electronic environment.

1. Internet Evolution

Progress in the field of information technologies is faster now than ever before. Browser services on the Web were created just 16 years ago, while the Google search engine is just 11 years old. Right now, a new tool that becomes popular throughout the world is found nearly every year. Millions of people use Facebook everyday to keep up with friends upload an unlimited number of photos, share links and videos, and learn more about the people they meet. This portal, for instance, was only created in 2004. YouTube, where videos can be posted, was established less than five years ago. Twitter – a real-time short messaging service – launched operations in 2006 and 2007. Radio and television audiences are asked to express their views on Twitter. Even e-mail is seen today as something that is "too slow".

There is no question that technologies offer advantages. It is so easy to publish information today that anyone who wants to express himself doing this. Grandmothers appreciate the Internet, because it allows them to communicate with grandchildren who are abroad. Small children first learn to read by using the computer.

Until the computer user encounters problems in the virtual environment, the ease of use is the most important aspect of technologies. People forget all about caution. Believe it or not, someone who committed a crime once videotaped himself doing so and then posted the video on YouTube! Of course, the police were quick to take advantage of the information. On social networks such as Facebook, in turn, information is increasingly being accessed and used by employers. A young mother who is proud of her firstborn and publishes the infant's life story from birth does not always think about how that will affect the child's future life. It is difficult to grasp the fact that once information appears on the Internet, it is there forever. Copies will be preserved somewhere, and you can never tell who will use that information and why. Parents really do need to monitor things their children post on the Net.

Yes, the Internet does contain a lot of educational content. Portals with any self-respect publish security instructions for its users. And yet often we forget that people only accumulate experience as to what should be protected in the electronic environment and how. We all know that we must keep track of our passport and our house keys, but we are only just learning how to protect our electronic identity and our passwords.

What to do, however, when a threat appears unannounced? There are situations in which a solution is impossible simply because the individual did not prepare in a timely way for the problems that might occur.

2. Computers and the Internet at Home

At the Eighth Workshop on the Economics of Information Security [8], there was the claim that small and medium companies nowadays have access to technologies that were available only to large enterprises 10 years ago. In Latvia, this thought can be carried even further – now, even households have access to such technologies. According to a 2008 study carried out by Latvia's Central Statistical Bureau, 63.7% of companies with fewer than 10 employees and 56.7% of the residents of Latvia were using computers, while 48.6% of the companies and 52.8% of the residents were also using the Internet [2]. We must also take into account human weaknesses, which include the desire for new "toys". This means that often people have newer and more powerful computers at home that they have at work, because companies need to think more about economic advantage.

Internet availability in Latvia reached the level of 58% in 2009, and among people who said they use the Internet, 86.6% had access at home [2]. Computer and, particularly, Internet use has become an inviolable part of everyday life in many households. People pay their bills, buy things, seek information and engage in the very necessary process of socialization by going online.

Moreover, 80% of respondents who had children living at home said they use the Internet [2].

Let's imagine a family of five to describe the solution that we need. We've got the married couple John and Kate, with teenage son Peter and six-year-old Emily. Also

grandmother Anna is living with them. According to statistics, that is exactly the type of family that is most likely to use the Internet most actively.

People of Latvia are most likely to use the Internet to send E-mail (83.7% of all users) and to find information (more than 70% of users) (Table 1), [2].

Table 1. Purposes for Internet usage by individuals, Year 2009

Purposes for Internet Usage	% of total population	% of Internet users
Sending / receiving e-mails	53.7	83.7
Telephoning over the Internet / video calls (via webcam) over the Internet	31.5	49.1
Posting messages to chat sites, blogs, newsgroups or on-line discussion forum	31.9	49.7
Finding information about goods or services	49.7	77.5
Listening to web radios or watching web television	30.7	47.9
Playing or downloading games, images, films or music	38.0	59.3
Reading or downloading online news / newspapers / news magazines	46.3	72.2
Internet Banking	42.2	65.8
Uploading self-created content (text, images, photos, videos, music etc.) to any website to be shared	33.8	52.7

The family that we’re considering uses several computers, including a powerful desk computer that can be used for computer games, a laptop for the teenage son, and a notebook computer that has been entrusted to John by his employer. It is sometimes used by the adults in the family. There’s a WiFi network in the house, and the router is hooked up to a broadband Internet connection. The family has a printer, several digital cameras, and some USB memory cards.

2.1. Threats against Household Computer Networks

The threats against household computer networks go beyond viruses and Trojan horses. Along with technological threats, there are also social and legal ones. There is also a separate issue of threats against data storage.

The best security solutions companies regularly offer information about the latest threats [9]. Example of this important information for information security experts is given in Table 2.

Table 2. Threats

Name	Type	Protected
Backdoor.Tidserv!gen9	Trojan	10/15/2010
W32.Pilleuz!gen13	Worm	10/14/2010
W32.Ackanta.H@mm	Virus, Worm	10/05/2010
Packed.Generic.308	Trojan, Virus, Worm	10/06/2010

Main technological threats are related to the connection of the home computer to the Internet and activities of users in the Internet. There is, of course, a possibility that there might be equipment failure, but the fact is that the greatest technological threat against home-based Internet users is malicious software. Sometimes its consequences can be relatively harmless (countless advertising messages, for instance), but in other occasions the software can be truly dangerous. Spyware, for instance, can be used to steal Internet banking data. The most common forms of malicious software at this time

are computer viruses, computer worms, Trojan horses, advertising software and spyware [10].

Technological threats occur mainly when not enough attention has been paid to the security of software. Major vulnerabilities include Internet browsers, Office software, E-mail programs, media players, and messaging software, all of which can be compromised.

Social threats on the Internet do not differ from those that are encountered on the street, at a store or in any other location. The problem is, however, that people often lack experience in using the Internet. Common threats on the Internet are the theft of personal data, cyber attacks, and computer games or gambling games. Each of these social risks bears with it a certain level of danger, but the most serious risk is entailed with identity theft, because it can create great financial losses or legal complications. There are also serious risks related to children and adolescents, including the risk of violence [11]. Additional risks in this area relate to activities in peer-to-peer networks [12].

When it comes to files that are kept in personal computers, there are two categories of data threats. The first is unauthorized access, which can be a problem for someone, for instance, who stores access codes to Internet banking systems on the computer. The leakage of such data can cause serious financial losses. There can also be deeply personal data which contain sensitive information. The bad news here is that it is very hard to note unauthorized copying of data. Household computers usually do not have the necessary control software, and the copying of data can be identified only when the data are actually used – i.e., when the harm has already been caused.

Legal threats at home can also relate to copyright violations. Copyright is protected by law in Latvia, and the primary duty is to protect authored works from unauthorized use. Software piracy is an issue that is covered here, as well.

Personal data in Latvia are protected by law, as well. The law on protecting the data of individual sets out a series of obligations for organizations which process personal data. Individuals have the right to do whatever they want with their own data, but the fact is that people do not always think about the fact that photographs from a party at home can contain the personal data of others, and such data must be handled in accordance with the law.

Criminal code of Latvia sets out regulations vis-a-vis various violations related to data processing systems. Criminal offences include unauthorized access to automated data processing systems, attempts to hinder the operations of such systems, and mishandling of information included in the systems. The owner of an improperly protected computer that is used to attack other computer systems is protected in part by the fact that investigators in Latvia have little experience in this area, work with investigators in other countries can be quite slow, etc. It is also true that it is very expensive to collect legally valid evidence in the Internet environment. That, however, should not suggest that anyone should be irresponsible when it comes to managing the security of a computer that is linked to the Internet.

Of course, this is not an exhaustive list of threats, and the fact is that the number and diversity of threats are increasing all the time. It must be added that there are also risks of a psychological nature – inappropriate perception of anonymous comments on the Internet, for example.

2.2. Everyday Management of Information Security

One area of endangerment that is not sufficiently understood at many households where there are Internet-linked computers is the organizational matter of who is responsible for what.

In the family that we are discussing here, for instance, the father is well-educated when it comes to protection against viruses. The tabletop computer at home has a high-quality anti-virus system, and it is updated regularly. The notebook computer that the teenager uses has an operating system which, according to general thinking, does not require an anti-virus system. The problem is that the teen has learned about reserve copying of data at school, and so he starts doing that at home in a less than systematic way, not analyzing whether he is really copying the most necessary data.

There is no panacea when it comes to information security, and no system is 100% secure. Even the systems of large banks and military organizations are not 100% protected against threats, even though these are systems in which vast organizational and technical resources have been invested. Specialists must work constantly in terms of reacting to new threats so as to maintain security at an appropriate level.

When it comes to corporations, good governance requires that responsibility for the security of information technologies is identified at the highest corporate level. Decisions related to information security must be based on the needs of the business and on an appropriate evaluation of risks. Information security management is both an organizational and a technical issue. There will always be residual risks of which management must be aware. They must be analyzed, and then relevant decisions must be taken.

Who, then, is responsible for information security management in a household? There is no law to dictate such responsibilities. Experience with best practice is shaped only over the course of time. And yet technologies are all around us now, and the number of people who use them is increasing very quickly, indeed. This means that the range of information risks is also expanding to include risks related to personal information.

3. Fundamentals of Information Security Management

Information systems security covers very diverse aspects, such as malicious code issues, security for operating systems and mobile technologies, encryption technologies and certificates management and many more.

The concept of “information system security” and the concept of “information security” are often used interchangeably. The word “system” is used most often when the issue relates to technologies. For this paper, by contrast, the concept of “information security” is more appropriate. The ISO 27002 standard defines information security as “preservation of confidentiality, integrity and availability of information; in addition, other properties, such as authenticity, accountability, non-repudiation, and reliability can also be involved”. The standard also states that “information security is the protection of information from a wide range of threats in order to ensure business continuity, minimize business risk”. Similar statements are contained in most other standards, and sources of information discuss information security as a process which relates to the protection of business information.

The above ISO standard determines that “an organizational data protection and privacy policy should be developed and implemented. This policy should be communicated to all persons involved in the processing of personal information”.

3.1. Process of Information Security

Business partners demand an acceptable level of information security one from another. Information security management standards should certainly play a major role in this regard. Various standards have been developed for organizing information systems security. The most popular standards are the ISO 27000 series of standards – for information security; the Control Objectives for Information and related Technology (COBIT) – set of best practices for IT management created by the ISACA and COBIT Security Baseline which includes 44 steps towards better information security; ISF Standard of Good Practice for Information Security [13].

In large organizations, information security management is ensured by qualified experts: “As security professionals work to elevate the level of security education and knowledge within their companies, one of the first hurdles is to reach a point at which organization members know what questions to ask and how to find the services they need” [14].

Too frequently, however, top managers at companies rely on information technology specialists without taking any responsibility themselves: “Even in organizations in which the security group has implemented a strong core program, it’s still challenging to get business units worldwide to take ownership of their security risks” [14]. There is a constant search for ways of speaking to managers about topics that are of personal interest to them: „Part of raising awareness involves personalizing risks for managers, showing them how vulnerabilities could affect them as individuals” [14].

Information security has to deal with more than just technologies. The relevant specialist must also have good communications skills: “more important that the organization has the right people to implement security successfully, meaning individuals who take ownership of security and build good relationships with others in the organization and external partners” [14].

The same applies to organizations: “One pivotal factor in creating a culture of security is setting the right “tone at the top”” [14]. As is the case with organizations, parents at home must define information security requirements for children and for themselves. This is a complicated issue, and not just in terms of the content of the rules that are put in place. When it comes to computer security, children are often smarter than their parents, and so the agreement might instead be based on an agreement as to roles and duties in this process.

3.2. Security Awareness

Several major organizations speak and write about security awareness. Since 2005 ENISA (the European Network and Information Security Agency) activities include also awareness raising tasks. The materials have been published [15], but their practical implementation has been left at the discretion of each country. The Agency has developed programs for Silver surfers, SMEs, Local government authorities and Media.

The Latvian project Netsafe is part of the European Union program Safer Internet plus and has a lot of success in promoting safer use of the Internet and new online

technologies, particularly for children. The Netsafe program is mainly targeted at educating school children; however, it also encourages the children and students to share their knowledge with their parents.

For several years already, OECD guidelines have listed information security among important subjects [16]. “Efforts to enhance the security of information systems and networks should be consistent with the values of a democratic society, particularly the need for an open and free flow of information and basic concerns for personal privacy.”. The guidelines describe 9 principles, with awareness and responsibility being ranked first among them.

Microsoft is one of the first to speak not only about the computer security but also about the personal safety of its users, i.e., people. This company has developed and published several educational materials.

The above materials provide diverse answers and advice how to protect one against different threats. However, these answers and advice are fragmented and deal with each threat separately. The full picture of information security management at home computer level is still missing.

4. Information Security Rules for Households

The home computer (Figure 1) is often perceived as a work station and people using it are treated as users. The technology dictionaries usually define user as a not very smart (also in terms of security) “element” of the system that performs simple operations, tries to circumvent security elements that disturb him/her, and consults a technology expert on everything that is not clear to him/her [17]. At home such a user also wants to relax and to access the social network as quickly as possible.

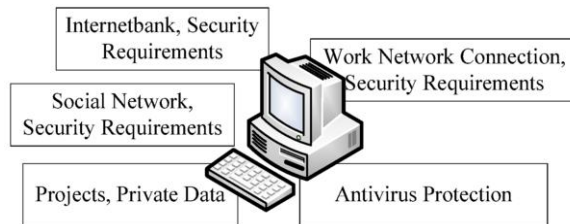


Figure 1. Home computer.

Quite frequently the home computer is operated by several people that are users of different information systems and, in a way, take care of the computer itself, its operating system, connection to computer network and security software. The traditional model treats those taking care of technologies and installing of security solutions as administrators. The administrator is the “element” who has the best knowledge of the system. A security administrator installs and manages security solutions and “fights” with the users. Home solutions sometimes imply expert assistance in installing software etc.; however, making of the daily backup copies remain a regular job for the user himself/herself.

These roles often overlap in a household. It is psychologically difficult for a person which is a “user” in his/her workplace to take responsibility for duties that are not day-to-day routine for him/her. Moreover, experts perform these tasks at workplace.

4.1. Security Policy

The goals of information security policy in households are to protect family information from unauthorized access. Here, policy issues include roles and responsibilities related to the management of information security in the family, monitoring of electronic information, and evaluation and classification of that information so as to determine whether it should be protected or published.

When it comes to policy planning in this area at home, people can certainly make use of information security standards that have proven themselves at work.

For companies, the recommendation is that information security management begins with management commitment: “ensuring an appropriate set of security controls is implemented enterprise-wide” [13]. At home, the first step is to understand that management of information security is also important in that environment. Latvian tradition says that men and women are responsible for their home and their yard. Countless folk songs praise orderly homeowners and mock those whose yards are never clean. This is a concept of responsibility that must also be transferred to the electronic environment.

Corporations also require information security policy to be properly documented. This is a principle that is also usable at home. The amount of information that is encountered every day is so vast that even the best and most friendly agreement can be forgotten if it is not written down.

The function of information security in terms of systematic planning and management of good practice in relation to information security could be entrusted to a teenager at home. Taking responsibility for information security at home is not much different, in truth, from taking responsibility for a pet. There are specific things that need to be done in a systematic way. What is more, this is an area in which teenagers often have a great deal of information, or at least a better chance to learn that information.

Mothers usually teach their children about safety on the street and elsewhere. Mothers can also take responsibility for security awareness in the family. There are various materials that will help – ENISA posters, but conversations and regular reminders are also very important.

4.2. Classification and Protection of Information

It is always important for companies to keep track of their assets and information. However, this aspect is sometimes underestimated at home. Before decisions can be taken such as to what kind of information requires what level of protection, the information must be registered (Table 3). This can be an interesting task for the whole family. When it comes to information that requires special protection, the threats must be evaluated, and the appropriate protections must be selected and put in place.

Table 3. Information classification

Information	Availability
Games	For all family members
School projects	Only for the owner
Travel pictures	Also to friends
Internetbank Access details	Only for the owner
Public notes	For everybody

We must not forget here about the physical protection of small items such as USB memory cards, cameras and mobile phones. For example, a mobile has turned into a device that is far more than just a communication tool. Mobile phones tend to store important information, and it is quite common for such information not to be copied for reserve purposes. Such items should never be left in a car or elsewhere where strangers can spot and perhaps steal them.

4.3. Roles and Responsibilities

Another very important element in information security policy is a list of roles and responsibilities (Table 4). Each of the duties must be described at a level of as much detail as possible.

Table 4. List of responsibilities

Responsibility	Who
Anti-virus management	John
Back-up	Peter
WiFi	John

To return to the issue of cleaning up your environment, it has to be said that a neat yard makes the entire neighborhood nicer, and each “neat” or securely management family computer network makes the overall virtual environment more secure and nicer, as well.

5. Risk Assessment Method

The best practice of information security management treats risk assessment as a key element that facilitates choosing of most appropriate protection solutions. However, attitudes towards risks or risk tolerances in private environment are very diverse; therefore there is no one-size-fits-all solution in this environment. Nevertheless, some protection solutions, such as antivirus software, are relevant for almost everybody. However, you can do without it if you avoid connecting your computer to the Internet or use operating systems where viruses are not common yet.

Until now the ordinary computer users have not been a target group for risk assessment methodologies. The author’s experience in educating users with different computer skill levels shows that there is a necessity for a solution that would help everybody to get relevant technological expertise in information security management.

Several extensive recommendations are available, e.g. by National Institute of Standards and Technology [18]. Before using them in a household, significant simplifying is necessary. The private environment is predisposed to intuitive use of different risk assessment methods building on previous experience. However, the experience in information system risks is tiny and it lacks systematization in most cases.

Risk management methodology is also a tool for structured discussions between business people and IT experts. Moreover, risk management methodology can also become a useful tool for raising the awareness of any user of electronic information.

The author builds her method for assessing information security risk on her almost ten years of experience in evaluating different kinds of IT-related risks and in using the NIST guide. The risk assessment method has been adapted for household environment.

The methodology for private environment focuses on a system for processing electronic information that comprises computers, networks and portable devices. Risk has been defined as the negative consequences of exploiting vulnerabilities taking into account the probability and the effects of an event. Risk management is the process of risk identification, assessment and reduction to an appropriate level.

The aim of managing risks in a private environment is to avoid hurting oneself by losing or over-disclosing private information due to lack of experience. From common network security perspective, it is essential to have reasonable protection for each computer so that it could not be used for malicious purposes. All household members, including head of the family, technologically most advanced family members and each information owner, have to share the responsibility for risk management.

The risk assessment is the first stage of the risk management methodology. The results of this stage help to determine the necessary controls. Taking into account household experience in relation to potential threats and vulnerabilities, the methodology should focus on the system description and the control awareness. Threat and vulnerability analysis and probability definition should be based on available statistical data or equivalent information.

5.1. Risk Assessment Process

Risk determination and control recommendations should be made in the form of an advice list. However, it is not necessary to document the results in a formal document. The table below summarizes the steps of a risk assessment in a household (Table 5) [18].

Table 5. Steps of risk assessment method

No	Step	Data source
1.	Description of electronic environment	Multiple choice questions, information on the equipment used, including the computer network and software, as well as on activities usually performed on the computer and/or in the Internet
2.	Identification of threats and vulnerabilities	Published data, analytical papers by experts in relation to drafting of a risk list
3.	Control awareness	Multiple choice questions, information on controls used, both technical controls and expertise
4.	Control analysis and probability determination	Information from steps 1 and 3, analytical papers by experts, calculations
5.	Influence analysis	Multiple choice questions about attitude towards risks, calculations
6.	Risk determination	Calculation, multiple choice questions about risk tolerance level
7.	Control recommendations	Information from step 6, recommendations by experts

Description of the electronic environment is the first step which is of vital importance. This step includes setting of the electronic environment boundaries and listing of hardware and software used, network connections, categories and purposes of information use, as well as the users.

Threat and vulnerability identification should build on both statistical data and on information gathered during the electronic environment survey. It should be noted that threat is not the same as risk. An example of a threat would be penetration of a virus into a computer. This threat can result in different risks, such as losing or damaging of data stored in the computer.

Experts ensure drafting of a list of potential risks and its improving following developments in this field. Some examples of risks are as follows:

- unwisely published data are used for stealing money;
- damage to the data stored in the computer due to virus operation;
- the computer becomes part of a botnet and is used for sending spam mail.

During the control awareness stage, multiple choice questions are used to account for controls used in the electronic environment, such as antivirus software, consultations by an IT expert, and adequately protected wireless network router.

Control analysis aims at analyzing the existing controls that reduce the probability of exposure to threats via vulnerabilities. Control analysis also comprises determination of probabilities. Controls in an electronic environment can be both technical and organizational.

Probabilities are determined by using data derived from statistics or other experience taking into account the description of the electronic environment, potential threats and control analysis.

Levels are used to depict probability measurements. Three levels are sufficient for a private environment: high, medium and low. High probability level means that a threat has a particular goal and controls for respective vulnerabilities are absent or of poor quality. Medium probability level denotes a situation when a threat is partially motivated and controls for respective vulnerabilities are almost adequate. Low probability level implies insignificant threat motivation and good controls. The result of the control analysis and probability determination stage is a probability level.

Influence analysis should describe the purposes of using the electronic environment, its criticality and sensitivity of stored information. This stage also implies use of multiple choice questions. Answer versions are prepared on the basis of statistical data on most common purposes of computer use in households. Similarly to probabilities, influence is also divided into levels. Three levels are sufficient for a private environment: high, medium and low.

Qualitative assessment is more appropriate for the private environment. Influence is high if the threat can affect the life of the household considerably. Influence is medium if a threat results in damaging or losing of significant resources. Influence is low if a threat results in a minor interruption in working in the electronic environment. Influence is determined for the original risk, i.e., without considering the control measures.

5.2. Risk Evaluation

The risk assessment stage aims at evaluation of the risk level. The methodology for the private environment uses a method from the NIST guide (Table 6).

High risk requires immediate measures for its reduction. Reduction is advisable in the case of a medium risk; however, one should evaluate the benefits against the expenses. Subjective opinion also plays a significant role in the private environment and determines the choice of additional controls that encourage subjective sense of protection. What regards to a low risk, usually the most adequate solution would be to accept it.

Table 6. Risk evaluation

Threat Likelihood	Impact		
	Low	Medium	High
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

The control recommendation stage aims at suggesting of solutions for reduction of the risk level. The final stage is acceptance of residual risks. Both control recommendations and residual risk explanations should build on simple and clear examples derived from daily life. They should contribute to overall information security awareness and encourage usage of adequate protection measures.

6. Conclusions

The goal of information security processes is to ensure that people have control over their computers and the information that is stored therein. When it comes to information security management at households, the first step is for each family member to understand that personal information on a computer and on the Internet can be endangered and that that means that proper protections must be put in place.

At the same time, each person will interpret the concept of “risk” in a different way. There are portals which allow one to post photographs. People do so. Then they forget about the images, but that doesn’t mean that the picture disappears. No, they travel all around the world. There are some people who are delighted about that. Others would not be happy at all to realize that an image of them is circulating around the globe. Of course, there are some people who are so interested in showing off that they think that the more people see their images, the better.

A second step in information security management at home is to understand what the personal electronic information is and to evaluate one’s approach towards how it is to be protected. There must be a differentiation among information that must never be published and must be protected carefully, information that does not require any particular protections, but is not published, as well as information that is to be published for as many viewers as possible. There are, of course, other levels of protection. Each family must decide on its own needs.

The third step (and only the third step) is to choose and install the relevant protections. Responsibility for the installation and maintenance of such tools can be divided up among family members. It is important to make sure that each member of the family is aware of his or her roles and duties in this regard.

Companies have financial tools which encourage their employees to keep information security requirements in mind: “Based on the results from an empirical study on the perceived cost and benefit of security measures, we concluded that the effort individuals are willing to make to comply with security policies is limited, and an organization can influence security behavior by managing the Compliance Budget effectively” [19].

But what about the household? People learn from their mistakes, but it would be best if well-organized information security management were to help people at home to avoid even the very first security problem.

Work on designing a tool for practical application of the risk assessment method continues. The tool is envisaged as an online application that uses data prepared in advance on both hardware and software used and on potential threats and risks. Some of the data are derived from statistics and research. The set of data used for the calculations will be supplemented taking into account threat developments and changes in user habits.

References

- [1] A global study of broadband quality September 2009, (available at <http://www.sbs.ox.ac.uk/newsandevents/Documents/Broadband%20Quality%20Study%202009%20Press%20Presentation%20%28final%29.pdf>)
- [2] Statistical databases: Information technologies, (available at <http://data.csb.gov.lv/DATABASEEN/zin/Annual%20statistical%20data/19.%20Information%20technologies/19.%20Information%20technologies.s.asp>)
- [3] Geon Woo Kim, Deok Gyu Lee, Jong Wook Han, Seung Hyun Lee, Sang Wook Kim (2009) Security technologies based on a home gateway for making smart homes secure, *Internet Research*, 2009, Vol. 19 Issue 2, pp. 209-226.
- [4] Brit ISP knocked offline by Latvian DDOS, (available at http://www.theregister.co.uk/2010/01/08/vispa_ddoa/)
- [5] Anderson, R. (2009) Information security: where computer science, economics and psychology meet, *Philosophical Transactions: Mathematical, Physical & Engineering Sciences*, July 2009, Vol. 367, Issue 1898, pp. 2717-2727.
- [6] Boyd, D. (2007) Why Youth (Heart) Social Network Sites: The Role of Networked Publics in Teenage Social Life. MacArthur Foundation Series on Digital Learning – Youth, Identity, and Digital Media Volume (ed. David Buckingham). Cambridge, MA: MIT Press.
- [7] Bryce, J., Klang, M. (2009) Young people, disclosure of personal information and online privacy: Control, choice and consequences, *Information Security Technical Report*, Aug2009, Vol. 14, Issue 3, pp. 160-166.
- [8] The Eighth Workshop on the Economics of Information Security (WEIS 2009), (available at <http://weis09.infoseccon.net/programme.html>)
- [9] Threat Explorer, (available at http://www.symantec.com/norton/security_response/threatexplorer/threats.jsp)
- [10] The Top Cyber Security Risks, (available at <http://www.sans.org/top-cyber-security-risks/?ref=top20#summary>)
- [11] Research 2008: "Safer Internet for Children and Youth", (available at <http://www.netsafe.lv/page/117>)
- [12] Johnson, E., McGuire, D., Willey, N. (2008) The Evolution of the Peer-to-Peer File Sharing Industry and the Security Risks for Users, *Proceedings of the 41st Hawaii International Conference on System Sciences – 2008*
- [13] ISF Standard of Good Practice for Information Security, (available at <https://www.isfsecuritystandard.com/SOGP07/index.htm>)
- [14] Johnson, E., Goetz, E. (2007) Embedding Information Security into the Organisation, *IEEE Security & Privacy*, May/June 2007, pp 16 – 24, (available at <http://mba.tuck.dartmouth.edu/digital/Research/AcademicPublications/SecurityOrg.pdf>)
- [15] Awareness Raising Quizzes Templates, (available at http://enisa.europa.eu/doc/pdf/deliverables/awareness_raising_quizzes_templates.pdf)
- [16] OECD Guidelines for the Security of Information Systems and Networks, *Towards a Culture of Security*, 2002, (available at <http://www.oecd.org/dataoecd/16/22/15582260.pdf>)
- [17] Tech Terms Computer Dictionary, (available at <http://www.techterms.com/definition/enduser>)
- [18] <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>
- [19] Beautement, A., Sasse, A., Wonham, M. (2008) The Compliance Budget: Managing Security Behaviour in Organisations, *New Security Paradigms Workshop*, (available at <http://www.nspw.org/proceedings/2008>)

Information Systems and AI Technologies

This page intentionally left blank

Random Spikes to Enhance Learning in Spiking Neural Networks

Janis ZUTERS¹

Faculty of Computing, University of Latvia, Latvia

Abstract. In spiking neural networks (SNNs), unlike traditional artificial neural networks, signals are propagated via a ‘pulse code’ instead of a ‘rate code’. This results in incorporating the time dimension into the network and thus theoretically ensures a higher computational power. The different principle of operation makes learning in SNNs complicated. In this paper, two ideas have been proposed to assure spiking activity propagation – random spikes and parallel input layers. The proposed ideas have been illustrated with experimental results.

Keywords. spiking activity propagation, random spikes, parallel input layers

Introduction

A neural network is a computational model which is made up of relatively simple interconnected processing units which are put into operation through a learning process. Neural networks are useful tools for solving many types of problems. These problems can be characterised as mapping (including pattern association and pattern classification), clustering, and constraint optimisation. There are several neural network models which are available for each type of problem [1].

Typically, the neurons of a network communicate via a ‘rate code’, i.e., by transmitting “pure” continuous values irrespective of the timings of the signals. Work with traditional models is usually arranged in steps – during each step exactly one pattern is classified or recognized. Considering this, any temporal information, if present in the problem, or interconnectedness of patterns is out of area of responsibility of a neural system, so it should be processed by a superior system.

By appearing of spiking neural networks, there also appeared a classification of neural network models into three generations.

The first generation of artificial neural networks consisted of McCulloch-Pitts threshold neurons [2], where a neuron sends a binary signal if the sum of its weighed incoming signals rises above a threshold value.

The second generation neurons use continuous activation functions instead of threshold functions to compute the output signals.

In contrast to classical models, spiking neural networks, or the third generation neural networks, are designed to make use of the temporal dynamics of neurons. Here, communication between neurons involves a ‘pulse code’ – the exact timing of the transmitted signal (i.e., the spike) is of key importance, but the strength of the signal is

¹ Janis Zuters, Faculty of Computing, University of Latvia, Raina bulvaris 19, LV-1586 Riga, Latvia; E-mail: janis.zuters@lu.lv

not considered. SNNs are believed to be very powerful, but their operation is more complicated and their application is not as obvious as is the case with “ordinary” neural networks. Nevertheless a lot of research has been focused on SNNs.

In [3], the method of ‘adaptive layer activation’ has been introduced for Hebbian-like learning in SNNs. The current paper further develops this approach (a) by adding injections of random spikes to induce the learning process, and (b) by extending network structure with parallel input layers of neurons in order to enrich presence of temporal templates in the input structure represented by axonal delays.

1. Spiking Neural Networks

Spiking neural network is a model of artificial neural networks, in which signals are propagated via spikes instead of rates (i.e., values). SNNs are potentially very powerful, still complicated to operate.

1.1. General Description

Each neuron in an SNN produces a train of spikes (i.e., it fires), and the operation of the neuron i is fully characterized by the set of firing times [4] (Figure 1):

$$F_i = \{t_i^{(1)}, \dots, t_i^{(n)}\}, \quad (1)$$

where $t_i^{(n)}$ is the most recent spike of neuron i .

One of the most frequently applied spiking neuron models is the ‘integrate-and-fire’ model [5][6]. In this model, input signals (both actual and recent) from a neuron are combined together in order to achieve enough activation for the neuron to fire.

Neuron i is said to fire if its activation state u_i reaches the threshold ϑ . At the moment when the threshold is crossed, a spike with the firing time $t_i^{(f)}$ is generated. Thus Eq. (2) can be clarified with Eq. (3) [4]:

$$F_i = \{t_i^{(f)}; 1 \leq f \leq n\} = \{t \mid u_i(t) = \vartheta\}, \quad (2)$$

where the activation state u_i is given by the linear superposition of all contributions:

$$u_i(t) = \eta_i(t - \hat{t}_i) + \sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in F_j} w_{ij} \varepsilon_{ij}(t - t_j^{(f)}), \quad (3)$$

where Γ_i is the set of presynaptic neurons to neuron i ; \hat{t}_i is the last spike of neuron i ; the function η_i takes care of refractoriness after a spike is emitted; the kernels ε_{ij} model the neurons response to presynaptic spikes.

$$\eta_i(s) = \begin{cases} -\vartheta \exp(-\frac{s}{\tau}); & s > 0 \\ 0; & s \leq 0 \end{cases}, \quad (4)$$

where τ is a time constant.

$$\varepsilon_{ij}(s) = \begin{cases} \exp(-\frac{s - \Delta_j^{ax}}{\tau_m}) - \exp(-\frac{s - \Delta_j^{ax}}{\tau_s}); & s - \Delta_j^{ax} > 0 \\ 0; & s - \Delta_j^{ax} \leq 0 \end{cases}, \quad (5)$$

where τ_s and τ_m are time constants, and Δ_j^{ax} is the axonal transmission delay for pre-synaptic neuron j .

Axonal transmission delay (or simply, axonal delay) is an essential part of a spiking neuron (Figure 1). It delays transmitted signals for the subsequent neurons, therefore it realizes a kind of short-time memory. Axonal delays are usually initialized in random with creation of an SNN.

Eqs. (4) and (5) have been adapted from [4], [7].

The pure integrate-and-fire model is very simple, but the fire sequences appear oversimplified and unable to reproduce rich and irregular nature of spiking of biological systems. In order to avoid regularities and deterministicities, numerous extensions of the model, as well as intermediate ones with other models have been proposed [8][9].

As using spikes makes a neural network to theoretically become more dynamic, stable activity propagation issues are of great importance [10].

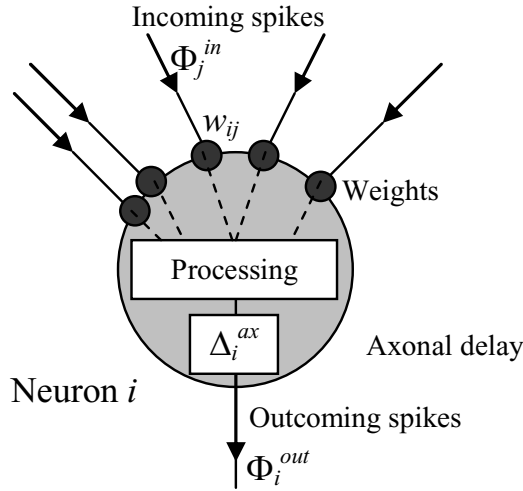


Figure 1. A single spiking neuron. The small filled circles w_{ij} denote synaptic weights. Φ_j^{in} denotes an incoming spike train j . The neuron produces the spike train Φ_i^{out} , delayed by axonal delay Δ_i^{ax} . (Adapted from [11])

1.2. Learning in Spiking Neural Networks

If supervised learning methods greatly predominate in “non-spiking” neural networks, then the situation with SNNs is quite different.

Back in 1949, Donald Hebb proposed a precise rule which might govern the synaptic changes which underlie learning: “When an axon of cell A is near enough to ex-

cite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A 's efficiency as one of the cells firing B , is increased" [12].

Operation of traditional neural networks is carried out through propagating of values, and the process of propagation is occurring without any constraints – the neural activities influence value strengths, while the propagation is occurring regardless these values.

$$\Delta w_{ij} = c_{ij} y_i y_j \cdot \quad (6)$$

Traditional neural networks use so called 'asynchronous propagation', where the propagation is occurring "automatically", without any conditions. The only issue then is the code itself to be propagated.

With SNNs, the things are different, because unlike traditional neural networks, there are conditions required for the process to occur, since spikes are generated only at certain conditions in a synchronous manner. So spiking activity propagation is a noticeable practical issue, which unfortunately has not received sufficient attention in literature.

This problem is already addressed by the author in [3] by proposing the method of adaptive layer activation.

The problem of spiking activity propagation is also addressed in [13] by comparing asynchronous and synchronous propagation modes.

2. Random Spikes and Parallel Input Layers

This research addresses the problem of propagation of spiking activity by proposing two ideas to ensure spiking activity propagation to occur.

- 1) Injecting random spikes to induce learning of an SNN;
- 2) Extending network structure with parallel input layers.

The proposed methods help to realize a way, how an SNN is set to operate. The experimental results show these methods to help to successfully overcome various problems to build SNNs.

2.1. The Learning Method Using Random Spikes

Applying effects of randomness is a method already used to extend integrate-and-fire neural networks [14].

This section introduces the approach of using random spikes in learning with SNNs. The current research is continuation of the previous work, where the following principle in operation of an SNN was assumed: learning within a neuron is performed only after the neuron has just fired, i.e., has just generated a spike. Since normally firing is accomplished only when activation of the neuron reaches a certain threshold value, the problem to overcome is – how to provide a neuron with conditions to frequently fire in order to undergo learning.

In the previous research, this problem was proposed to solve by using the 'activation acceleration factor'. The main idea of it is that the requirements of firing are being turned down, if neural activity in the layer is insufficient. Such a situation is typical at

the very beginning of the learning process. The approach of activation acceleration helps in most cases, still in cases when the signal is zero, no turning down the requirements allows us to avoid freezing.

In the new approach, this principle is substituted by generating random spikes with a certain probability. By this, a minimum activity of a neuron is guaranteed, and this ensures the learning process to happen.

Although the new method is simple for implementation, the effect of bringing the network into activity from it is noticeable.

The algorithm in Figure 2 shows the learning algorithm for a whole layer using this principle.

```

Algorithm random_spikes ( $t, \rho$ )
   $t$  – time step of operation
   $\rho$  – random spike probability ( $0 \leq \rho < 1$ )
   $\vartheta$  – threshold to be reached by activation of the neuron
   $\Delta^{\text{ax}}$  – axonal delay for the neuron
  run_spiking_neuron_random_spikes – algorithm to operate one neuron with ‘random spikes’
Begin
  Forall neurons  $i$  in the layer
    % (A) run the neuron
    Compute activation  $u$  by processing timings of incoming spikes and weights (Eq. (3))
     $\text{fire} \leftarrow \text{False}$ 
    If  $u \geq \vartheta$  Then % activation reaches threshold
       $\text{fire} \leftarrow \text{True}$ 
    Elseif get_random_value  $< \rho$  Then % probability to generate random spike
       $\text{fire} \leftarrow \text{True}$ 
    If  $\text{fire} \geq \vartheta$  Then
      Generate spike at time step  $t$ 
      Delay the spike by  $\Delta^{\text{ax}}$  to become ‘incoming’ spike for subsequent neurons
    % (B) adjust weights
    If neuron  $i$  has just fired Then
      Forall weights  $j$  in the neuron
        Increase weight  $j$  proportionally to the respective incoming activation (Eq. (6))
        Decrease uniformly all the weights to keep a fixed average weight value of the neuron

```

Figure 2. Learning with the method of ‘random spikes’ for one layer

2.2. Using Parallel Input Layers in Network Structure

This section proposes an SNN architecture with parallel input layers to enhance the ability of a neural network to detect regularities in an input.

Commonly a neural network has a linear feed-forward structure. This also holds for SNNs (Figure 3). For technical reasons, ‘input layers’ are usually not “true” layers: input values to the whole neural network are directly put to the outputs of the input layers, so they don’t compute anything.

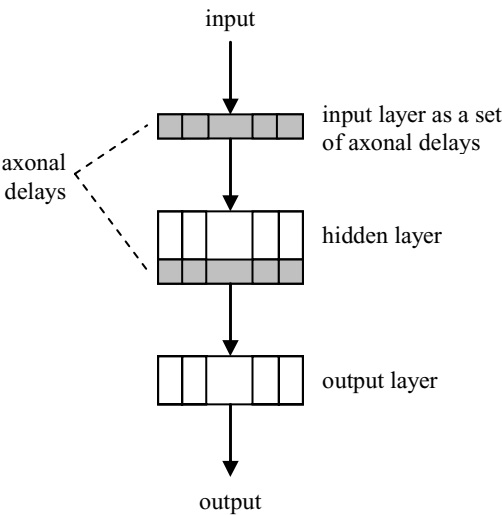


Figure 3. A simple example of a feed-forward linear architecture of a spiking neural network. Connections between layers are depicted as arrows, and they denote all-to-all connection between the neurons of both layers

With SNNs, the input layer can additionally have the role of delaying input signals. As axonal delays are predefined in random, input layers of SNNs provide with a kind of spatial representation of temporal input information.

Instead of a single input layer, several parallel input layers are proposed by the author. Since axonal delays of neurons are predefined random values, bunch of input layers provide with several kinds of spatial representations of input signals. (Figure 4)

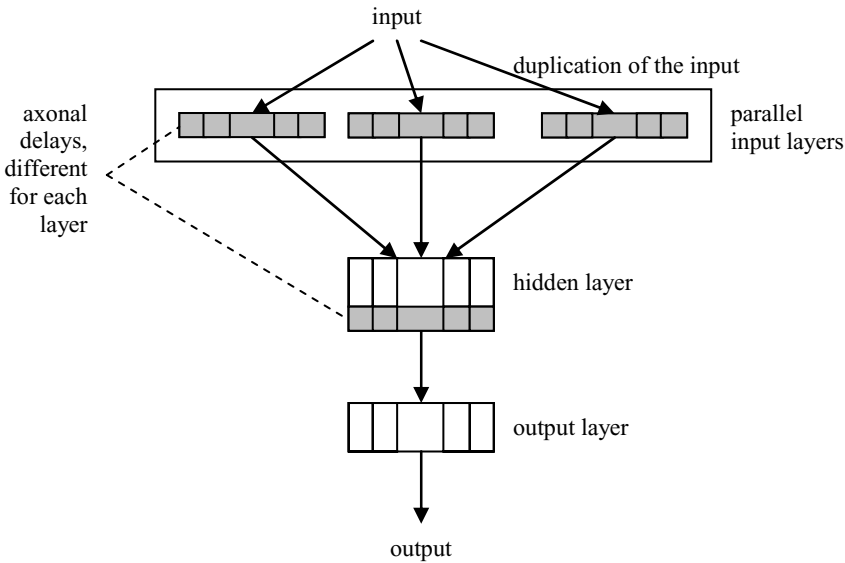


Figure 4. A spiking neural network architecture with parallel input layers

This approach is also about spike activity propagation, because since the input signal is duplicated by some changing, we get more configurations of signals to propagate to the next layer, so we get more chance to meet propagation conditions for the next layer.

The idea is very simple: at each iteration, the input is concurrently passed to all the input layers (see the algorithm in Figure 5).

Algorithm run_neural_network_parallel_input_layers

Begin

Do many times

Fetch next input into X

Forall L in input layers **Do**

 Set X to L

 Process the rest of layers in a row

Figure 5. Running a network with parallel input layers

By this, the next layer of the network receives several varieties of partly delayed input signals, so parallel input layers serve as a grid of first level perception.

As the experimental results show, this leads to faster and better learning in an SNN.

3. Experimental Work

To examine the proposed approach, a large experimental work was accomplished using original software for that. Each experiment was run to solve a task of detecting one kind of temporal patterns in an input stream (in an unsupervised manner). Experiments were carried out using several configurations of SNNs to prove the concept in ensuring better results.

3.1. Setup of the Experiments

The proposed learning method was tested on the detection of reiterative temporal patterns in an input stream. A trained neural network should be able to detect such patterns by giving a spike in output after the pattern is encountered. Initially, no supervised learning was performed to the network, so the network didn't know which kind of patterns to look for. So the network was to determine itself what to seek.

The testing environment produces three binary signals at each discrete time moment t and passes them to the input of the neural network. In general, signals are generated at random, but sometimes patterns from a predefined store are interwoven into them. There are six possible temporal patterns in the store (Figure 6). The proportion of 1s and 0s in the random part equals the one in the patterns – 20%. So, in the input stream, the patterns are easy to (visually) observe (Figure 7) while the proportion of 1s and 0s remains constant.

At each separate experiment, one of the six patterns was occasionally put into input stream (7 patterns of size 10 within 200 time steps in average), so the proportion of patterns in the input stream was $7 \times 10 \div 200 = 35\%$.

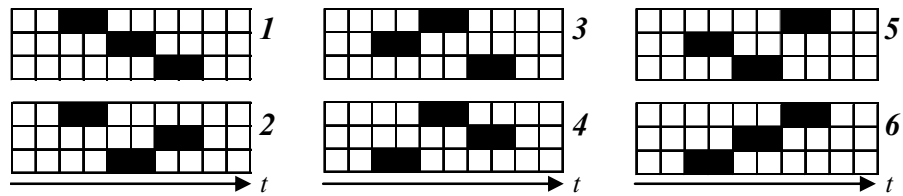


Figure 6. The 6 temporal patterns used in experiments: width=3 signals, length=10 times; black squares denote ‘1’ or ‘spike’, but white squares – ‘0’ or ‘no spike’

Output spikes of trained networks were analyzed of how they match patterns in the input stream. Although the predefined patterns was of constant size, it was not predefined at which exact position a pattern should be detected (it could differ from experiment to experiment, however it should be fixed within one experiment). Pattern recognition rate was measured as a rate of responses of a network to input patterns at a fixed position with regard to the end of a pattern. Positions tried were of the interval $[-5, +5]$. As for instance, pattern recognition rate of 90% means that we have a position p , such one, for which is true that for 90% of patterns of the input stream, a spike of the neural network is encountered at the position p with regard to patterns. In addition, the rest of spikes, false positive ones, also were counted (Figure 7.). False positive rate refers to the proportion between false positive spikes and actual patterns in the input stream.

To show the introduced methods working, a total of 15,000 experiments were conducted. Each experiment consisted of 20,000 iterations in the learning phase, and then 200 iterations in the recall phase, the results of which were recorded and further analyzed.

Five different configurations of SNNs were used (Table 1) according with the general structure shown in Figure 4 and the algorithm in Figure 2. Different configurations of SNNs were used to be able to compare them in order to mark out the effect of the two ideas introduced in this paper: random spikes and parallel input layers.

The functionality of SNNs was built as described in sections 1.1 and 2; with random spike probability – 4%, threshold – 0.3, and axonal delays were initially randomly set from the interval $[0, 4]$.

Along analyzing direct results of the experiments, also side affects were determined, those of ‘false positives’ and position of pattern detection.

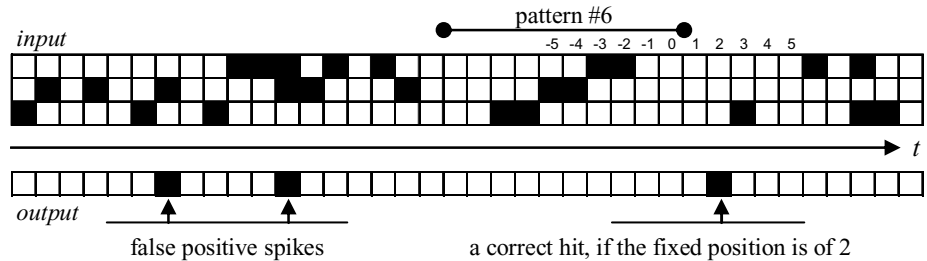


Figure 7. An excerpt of an input stream, and examples of responses of the network to it. Here the value of position can be chosen as 2, so the pattern recognition rate is computed as 100% (as the only pattern is recognized), and the false positive rate is 2 (two false positive spikes against one pattern).

Table 1. Network configurations used in the experimentation.

Name of the configuration	Usage of random spikes	Network architecture description (for all types, there are 3 inputs and 1 output)
A. Random=N 4×3-12-1	No	4 parallel inputs, 12 hidden neurons
B. Random=N 8×3-24-1	No	8 parallel inputs, 24 hidden neurons
C. Random=Y 4×3-12-1	Yes	4 parallel inputs, 12 hidden neurons
D. Random=Y 1×3-24-1	Yes	no parallel inputs, 24 hidden neurons
E. Random=Y 8×3-24-1	Yes	8 parallel inputs, 24 hidden neurons

3.2. Analysis of the Results

Figure 8 shows the general overview of experimental results for the five configurations of SNNs.

The configurations C, D, and E show a positive effect of using random spikes to recognition.

The configuration E shows the impact of additional parallel input layers, while D and E show the effect of both parallel input layers and a sufficiently sized hidden layer.

The configuration D (in contrast to C) might signal a large hidden layer to bring the same effect as parallel input layers do, however it suffers from instability of detection (see Figure 10).

However it was not predefined, at which position a pattern should be detected, the two things are preferred in order to keep the process stable:

- 1) The position should be close to the end;
- 2) Keep position changes among experiments as small as possible.

Also with regard to the 'position stability', the configuration E is the best.

To summarize the experimental outcomes, the best results were obtained by obeying all the following conditions:

- 1) Using random spikes;
- 2) Using more parallel input layers;
- 3) Using a hidden layer with a sufficient size.

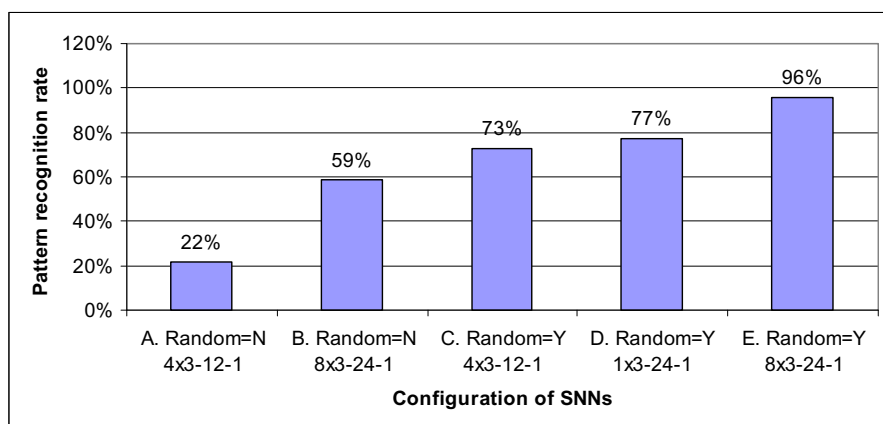


Figure 8. Pattern recognition rate obtained for various configurations of SNNs. For the best results, random spikes and parallel input layers should be used

The main results demonstrate SNNs to be successfully trained to detect temporal patterns. During experiments, there also were undesirable side effects, which should be reduced in further research.

The most noticeable issue here is that of false positives (Figure 9). The results show that, along successful detection of input patterns, a lot of redundant spikes are additionally generated by a neural network. Although pattern recognition rate is rather high, i.e., patterns are detected precisely, there are more redundant spikes than expected ones.

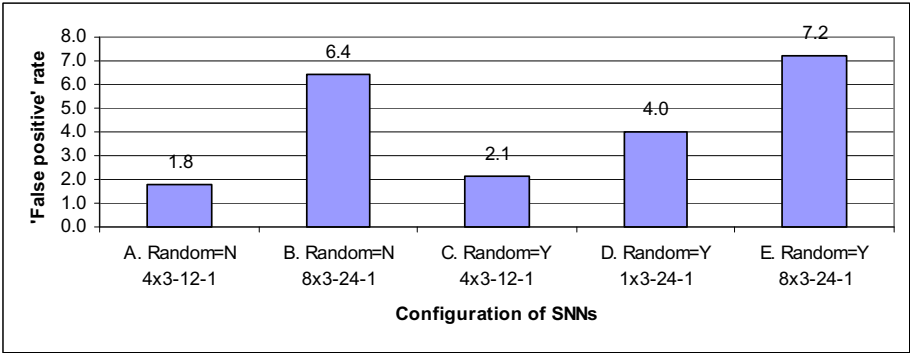


Figure 9. High ‘false positive’ rates signal the algorithms to be further improved and/or better configured/setup. False positive rate refers to the proportion between false positive spikes and actual patterns in the input stream

Figure 10 shows how the architecture of the network influences the length of detected patterns.

The networks of the configurations C and D detect most patterns at earlier positions (See also Figure 7), while the stronger networks (E) typically detect patterns almost at the end position (-1), moreover the position of detection was much more determined (50% for the positions of highest occurrence). The configuration D is especially unstable, so it has such a good recognition rate (Figure 8) just because of the methodology used.

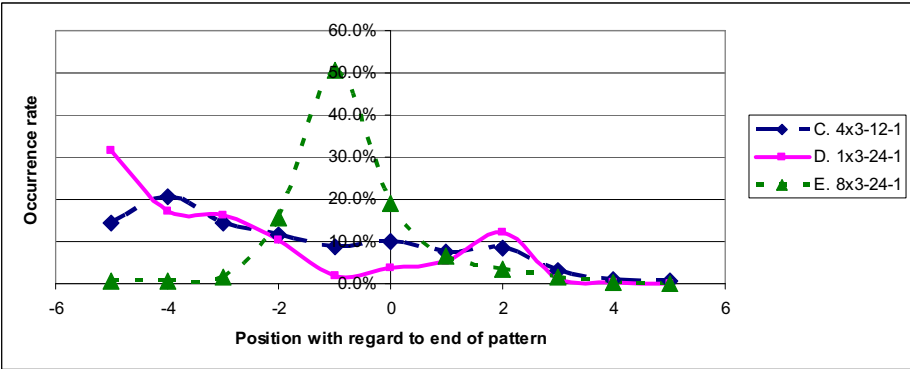


Figure 10. ‘Position stability’. Occurrence distribution of positions (with regard to the ends of patterns) at which the best pattern recognition rate was obtained (only for the three configurations with random spikes). The configuration E shows the most stable results

4. Conclusion

Spiking neural networks are believed to be very powerful. For all that they are comparatively complicated, especially in terms of learning.

One of the issues with spiking neural networks is that of spiking activity propagation, i.e. ensuring a continuous operation of an SNN. In this paper, two ideas have been introduced to address this:

- 1) Using random spikes to additionally stimulate the neuron;
- 2) Using several parallel input layers to better preprocess the input.

The principle of random spikes ensures the minimum activity of a neuron, and by this, also the learning process to happen. Experimental results show it essential for a good recognition rate.

Although the experiments confirm the significance of hidden layers with sufficient sizes, yet for the best results, parallel input layers are very important to have.

With the use of proposed approaches, some side effects should be also considered: position stability and false positive spikes. In order to overcome these, there should be put some additional effort with the practical implementation.

References

- [1] L. Fausett. *Fundamentals of Neural Networks*. Prentice-Hall, Inc., 1993.
- [2] W. Maas. Networks of spiking neurons: the third generation of neural network models. In: *Transactions of the Society for Computer Simulation International*, vol. 14, issue 4 (December 1997), pp. 1659-1671, 1997.
- [3] J. Zuters. Spiking Neural Networks to Detect Temporal Patterns. In: Haav, H.M., Kalja A. (eds.) *Frontiers in Artificial Intelligence and Applications*, vol. 187, Databases and Information Systems V - Selected Papers from the Eighth International Baltic Conference, DB&IS 2008, pp. 131-142, 2009.
- [4] W. Gerstner. Spiking Neurons. In W. Maass and C. M. Bishop (Eds.), *Pulsed neural networks*. MIT Press, 1999.
- [5] A. N. Burkitt. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. In: *Biological Cybernetics*, Volume 95, Number 1, 2006, pp. 1-19. Springer, 2006.
- [6] A. N. Burkitt. A review of the integrate-and-fire neuron model: I. Inhomogeneous synaptic input and network properties. In: *Biological Cybernetics*, Volume 95, Number 2, 2006, pp. 97-112. Springer, 2006.
- [7] W. Maas. Computing with Spiking Neurons. In W. Maass and C. M. Bishop (Eds.), *Pulsed neural networks*. MIT Press, 1999.
- [8] E. M. Izhikevich. Simple Model of Spiking Neuron. In: *IEEE Transactions on Neural Networks*, Volume 14, Number 6, November 2003. IEEE, 2003.
- [9] M. Salerno, D. Casali, G. Constantini, M. Carota. Fully Asynchronous Neural Paradigm. In: *Proceedings of the 15th IEEE Mediterranean Electrotechnical Conference (MELECON 2010)*, pp. 1039-1043. IEEE, 2010.
- [10] W. M. Kistler, W. Gerstner. Stable Propagation of Activity Pulses in Populations of Spiking Neurons. *Neural Computation* 14, 987-997. Massachusetts Institute of Technology, 2002.
- [11] R. Kempter, W. Gerstner, J., and L. van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59:4498-4514, 1999.
- [12] D. Hebb. *The organization of behavior*. John Wiley and Sons, New York, 1949.
- [13] A. Kumar, S. Rotter, A. Aertsen. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. *Nature Reviews Neuroscience* 11, 615-627 (September 2010), 2010.
- [14] H. S. Seung. Learning in Spiking Neural Networks by Reinforcement of Stochastic Synaptic Transmission. In: *Neuron*, Volume 40, Number 6, 18 December 2003, pp. 1063-1073. Elsevier, 2003.

Interactive Inductive Learning System

Ilze BIRZNIECE

Riga Technical University, Department of System Theory and Design,

Kalku 1, LV – 1658, Riga, Latvia

ilze.birzniece@rtu.lv

Abstract. Inductive learning system learns classification from training examples and uses induced rules for classifying new instances. If a decision cannot be inferred from system rule base, a default rule is usually applied. In the paper a new interactive approach is proposed where in uncertain conditions an interactive inductive learning system can ask for human decision and improve its knowledge base with the rule derived from this decision. Problems and solutions of incorporation of human-made decision into rule base and aspects of choosing between static and incremental learning algorithms are analyzed in context of the proposed approach. An interactive inductive learning system is proposed to assist in study course comparative analysis.

Keywords. Inductive learning, interactive inductive learning, machine learning, human-computer interaction, course comparison

Introduction

The ever-growing importance of machine learning in multiple problem domains has been highlighted in many articles, e.g. [1,2,3,4,5, and 6]. In general, machine learning algorithms are domain independent. However, there are domains where not only a high predictive accuracy, but also interpretability of generated descriptions is essential. Inductive learning algorithms are widely used in machine learning tasks and they hold a strong position as reliable classification methods that can explain their decision making process [3]. Inductive learning methods are considered attractive for many real-life applications (e.g., medical diagnostic [7] and industrial visual inspection [8]). There is still potential to improve the performance of inductive learning and to extend its area of application. One of the problems to be solved in inductive learning is inability of classifier to always derive the class membership for a new instance using only its existing knowledge base.

One of the domains aside from the majority of typical problem areas that can benefit from machine learning is curriculum management. The necessity to compare different study courses leads to the course classification system that learns from expert's comparison of different study courses.

The goals of this paper are the following: (1) to show the need for an additional approach in dealing with instances that can't be classified using induced rules set; (2) to prove that non-classifiable instances can be handled with the human help; (3) to explain the procedures of human-classified instance incorporation into existing rule base; (4) to define study course comparison as a problem to be solved with the proposed interactive inductive learning system.

1. Division of Inductive Learning Methods

An explanation of the division in static and incremental inductive learning algorithms (Section 1.1) is necessary to clarify the methods proposed in Section 3. The need for multi-label classification (Section 1.2) will concern in context of problem domain addressed in Sections 5 and 6.

In context of inductive learning two terms – “example” and “instance” - are used in this paper. Although they both refer to a separate data unit described by particular attributes and values, their meanings are specified in the context of the paper. By using the term “example”, a unit for classifier training is meant (item in training set), whereas “instance” denotes a new unit to classify with no classification known. In other words, the observed unit in the classifier training stage is called “example” while a unit in the classifier applying stage is called “instance”. Instance can turn into example if it is classified and added to training set.

1.1. Static and Incremental Inductive Learning Methods

Depending on the kind of learning, inductive learning methods can be divided in incremental and nonincremental (or static) ones. In context with the problem addressed in this paper it is worth going into detail with these two types of learning algorithms. According to [9], static algorithms are appropriate for learning tasks where a single fixed set of training examples is provided while incremental algorithms revise the current concept definition in response to each new training example.

In real-world domains there are two problems known as hidden context and concept drift [10]. Hidden context means that target concept may depend on variables, which are not given as attributes. When changes in hidden context induce changes in target context the problem of concept drift appears. These problems are discussed in more detail in [10, 11]. Incremental learning has the facility to adapt to changes in the target concept when creating classifier for real-world data streams [10]. Such ability is hindered or even impossible for non-incremental learning methods. Incremental learning algorithms can be divided in three groups depending on example memory they have.

- *Full example memory* stores all training examples, which allows for efficient classifier restructuring and good accuracy, but it needs huge storage. Such algorithms are ID5, ID5R, ITI [9, 11].
- *No example memory* stores only statistical information and thus loses accuracy but saves storage space. Examples of this group are ID4, STAGGER, AQ11 algorithms [11].
- *Partial example memory* only stores selected examples, which is a compromise between storage space and accuracy. The most popular methods with partial example memory are HILLARY, FLORA, AQ-PM [10,11].

One of the reasons that motivate to study and implement learners with partial example memory is awareness of the fact that “humans not only store concept descriptions, but also remember specific events” [11]. An inductive learning system has to choose these specific events. For partial example memory there are several techniques used for choosing examples (events) to store [12]. All these techniques have problems to be solved to maintain classifier accuracy. No such problems appear with

static learning algorithms, so it may not be worth to implement an incremental algorithm for a constant or slowly changing learning set. The most popular static learning algorithms are ID3, C4.5, CART, AQ, and CN2 [2,4].

1.2. Single-Label and Multi-Label Classification Methods

In traditional single-label classification examples are associated with a single label l from L , $|L| > 1$, where L is a set of disjoint labels. In multi-label classification (MLC) each example may be associated with a set of labels $Y \subseteq L$. The majority of classification approaches do not consider creation of possibly useful rules with multiple labels. A single-label classification algorithm tries to decide between potential rules (if there is more than one corresponding rule) when it classifies new instance and extracts only one class associated with the most obvious rule.

However, MLC rules may often be useful in practice, since they bring up useful information that also has relevance and otherwise ignored rules may play a role in prediction and be very useful to the decision maker [13]. The need for MLC shows up in fields such as bioinformatics, scene classification and text categorization.

Processing of multi-label data can be done in two ways: a) problem transformation, and b) algorithm adaptation. Problem transformation methods are algorithm independent. They transform the learning task into one or more single-label classification tasks, for which a large number of learning algorithms exist. Algorithm adaptation methods extend specific learning algorithms in order to handle multi-label data directly. More information about multi-label learning can be found in [13, 14, 15, 16, 17].

2. Background in Dealing with Non-Classifiable Instances

When applying classifier it may happen that none of the rules in the rule base (or leaf in the tree) fits the new instance. It is possible that some unique or exceptional instance is encountered, or rules do not cover all areas in problem domain.

There are several methods to deal with this problem. AQ and CN2 algorithms apply a default rule that predicts the most common class in a particular data set if none of the generated rules fits the instance [18]. This is the most common approach in dealing with non-classifiable instances. However, it does not work well in all domains. If a data set contains many classes and, moreover, if all of them occur equally frequently, assigning one certain class to all unclassified instances will not lead to high accuracy of classifier. It is evident, that using a default rule in domain with 50 classes, which occur in similar frequency, could lead to 2% average classification accuracy for non-classifiable instances with every 50th instance classified correctly.

Within the domains where it is more critical to detect one class from among a group of classes, e.g., medical diagnosis, this most relevant class is usually assigned, when classification can't be made, so as not to make a serious mistake. Yet such a method is not appropriate for all situations. Unbalanced data sets, where one class occurs much more frequently than another, often appear in many practical applications and it is important to deal with classification as accurate as possible also in these domains [19].

In [10] incremental rule learning based on example nearness from numerical data streams is presented. This learning system uses two types of rules. Consistent rules

classify new instances by covering them and inconsistent rules classify instances by distance as in the nearest neighbour algorithm. Instances not covered by any rule are classified based on the label associated with the reliable rule that has the smallest distance. This approach is specific and cannot work with nominal attributes where nearness can't be calculated.

Summarising existing approaches, one can assume that if in the traditional process of classification no rule that can classify a particular instance is found, a guess is made. Most frequently this "guess" means the default rule.

As classification tasks are getting more complicated, the computer program often meets situations where classification can't be done with existing rule base. In situations where new instances can't be unequivocally classified, a collaborative approach between the machine and system user (human expert) would be useful. The paper proposes to construct a new inductive learning system that could ask a human to make classification in the above-mentioned situation and improve its knowledge base with a rule derived from this experience.

3. The Proposed Interactive Inductive Learning System

Survey of related work on interactive inductive learning and different types of human involvement in classifier training and applying can be found in [12, 20]. None of the methods described in previous related work provide an appropriate model of interactivity for solving the inductive learning problem with classifying instances that do not fit any rules in knowledge base. As stated in Section 2, current approaches to this problem do not work well in all situations. Therefore a new computer-human interactivity model for dealing with non-classifiable instances approach is proposed.

The proposed system involves a human when it meets a new instance not consistent with rule base, thereby replacing a default rule. The whole cycle of the interactive inductive learning system involving a human is depicted in Figure 1.

The structure of the system tends to provide mutual independence for both the classifier and the human. This position involves the presence of the following properties.

- System is not dependent on human; basically it operates by itself.
- Human does not have to answer system's questions constantly.

More benefits can be gained from expert knowledge besides the correct classification of every single instance presented to the expert. The human decision can also be used to improve rule base since human-given advice can be saved and formed as a new rule to be incorporated into existing rule base.

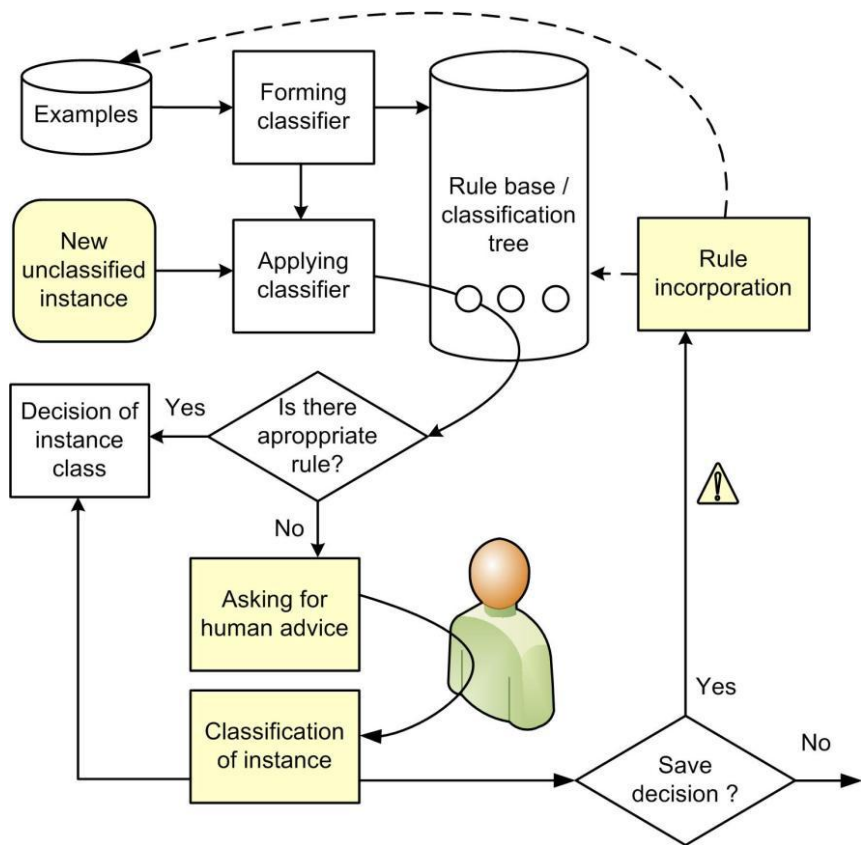


Figure 1. Classification with human interaction

4. Adoption of Human Decision

Questions concerning maintaining rule base consistency in an interactive inductive learning system are discussed in [12]. The task of interactive algorithm is to accept human decision, and to update classifier in response to this decision. Different learning strategies with varied consequences can be used to process a human classified example. The essential question to discuss is whether to treat the human classified instance already as a rule or just as a new training example which is known to be correct. Consequences from this decision are depicted in Figure 2.

If the human decision is treated as a rule, the question arises how to clarify the length and essential attributes of this rule. It is worth to consider a case where many attributes describe the instance. Storing the rule with all attributes and values that are represented in an instance is ineffective in both from memory space and generalization viewpoint. In longer term this approach could reduce classifier’s ability to generalize, thus decreasing predictive accuracy.

If a new classified instance is treated as a new example to learn from, further actions depend on learning strategy chosen for classifier. In case a static inductive learning method is used to train classifier, the new example is enclosed in a training set

and classifier should be created from scratch. Such approach is time and computational resource consuming and ineffective comparing to incremental learning methods if instances are presented serially [9]. This may not be the critical judgement as long as the proposed interactive learning system is not offered as a real-time learning system. Obviously it depends on the particular domain and learning task. A step towards computationally more effective classifier rebuilding would be setting the threshold and waiting for more examples to come and only then incorporating examples into classifier when the limit of instance count is reached. Until this limit is reached instances could be stored as rules, thus merging both opportunities – temporary treating the human decision as rule and incorporating it within training examples in order to generalize it afterwards.

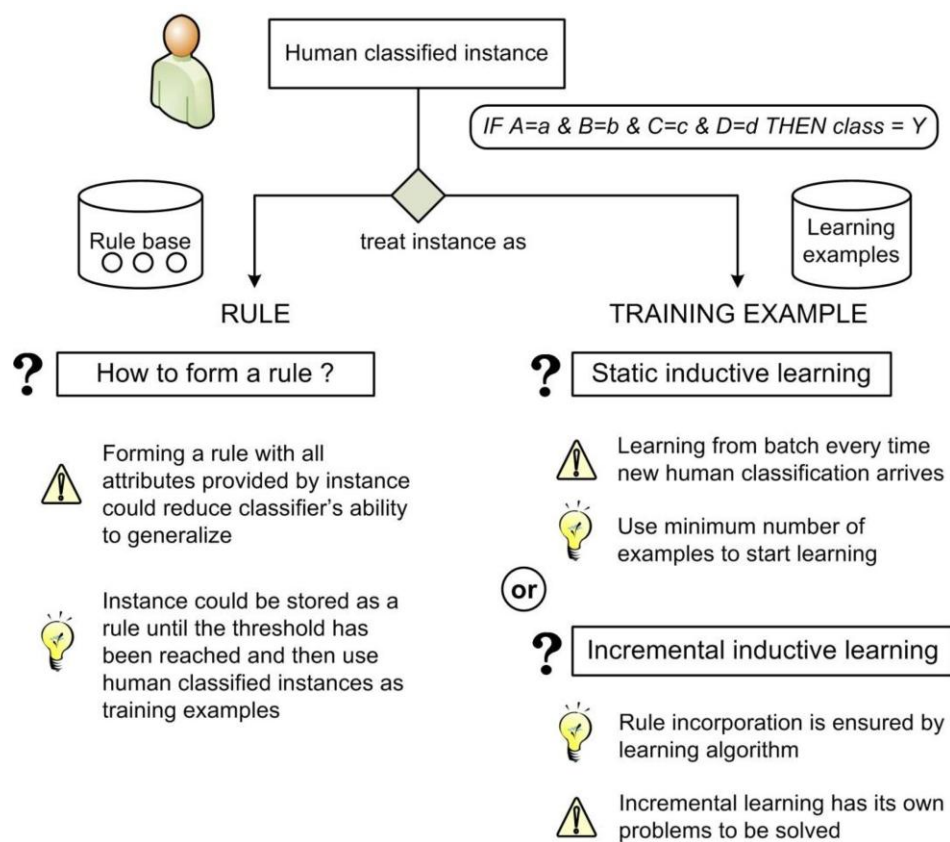


Figure 2. Possible options and problems with human classified instance incorporation

Finding the amount of examples for the threshold is a subject of further studies and most likely will depend on (1) the time needed to build classifier from the initial training set and (2) frequency of human classified instance appearance. The higher the costs for learning from batch, the higher the minimum of examples limit will be set.

Use of the incremental learning method has its advantages and drawbacks. The use of the incremental method is desirable because incorporation of human-made decision and rule base integrity will be under competence of the learning algorithm. On the

other hand, by using incremental learning algorithms the open question of examples' memory size has to be solved [11].

Considering options from Figure 2, the following preferable approaches for human knowledge incorporation into existing rules set emerge.

Threshold based static learning approach that includes the following steps:

- 1. Set a threshold - positive number, representing the number of instances classified by the user.
- 2. Store human classified instance as a rule with all attributes and their values unless the threshold has been reached;
- 3. Use static inductive learning method to rebuild the classifier and include human classified instances into training base;
- 4. Remove human classified instances used so far as rules from rule base;
- 5. Replace classifier with the new one.

Incremental learning approach that includes the following steps:

- 1. Add human classified instance directly to the learning algorithm and let it be included in classifier as defined by a specific inductive learning algorithm;
- 2. Use updated classifier.

As described in Section 1.1, incremental inductive learning methods may be divided into algorithms without example memory, with partial example memory and with full example memory. Table 1 shows strong (+) and weak (-) sides of inductive learning types considering classification environment where human classified instances arrive occasionally. Symbol “+” denotes higher accuracy, smaller relearning costs, smaller storage costs, and better acceptance of concept drift. No detailed comparison has been given because explicit characteristics may depend on the algorithm used. Table 1 summarizes data from [9, 10, 11, 21, 22].

Table 1. Characteristics of different inductive learning types

	Static	No example mem.	Partial example mem.	Full example mem.
Accuracy	+	-	-	+
Relearning costs	-	+	+	+
Storing costs	-	+	+	-
Concept drift	-	+	+	-

According to [11], the static learning approach could lead to a higher accuracy of classifier, compared to the incremental approach using partial or no examples' memory.

If the expected frequency of additional instance arrival is high, then an incremental learning algorithm should be considered. Depending on storage amount, desired accuracy, and other considerations, the method with full, partial or no examples' memory is to be chosen.

The choice of learning strategy becomes an optimization task where predictive accuracy, relearning costs, storing costs, and other relevant characteristics should be weighted. There are no context and domain independent reasons to define one learning method to be better than the others [23]. The choice of a particular method depends on the type of problem, requirements, constraints, background knowledge, and many other issues.

The following subsection provides an example of new rule incorporation into existing rule base with the help of a threshold based static learning approach.

Incremental learning approach isn't demonstrated because there are many different incremental methods which operate with new instances according to their algorithms, whereas a threshold based static learning approach has not been described before.

4.1. An Example

With the following elementary example the operation of a threshold based static learning approach will be demonstrated. Table 2 shows initial training examples for classifier forming in domain of natural phenomena for season prediction. There are three nominal attributes, namely, temperature, leaf status of trees, and weather. Class label determines the season.

Table 2. Training examples for season prediction

	Temperature	Trees	Weather	Season
1	moderate	yellow	rainy	autumn
2	moderate	bare	sunny	spring
3	high	green	sunny	summer
4	low	bare	sunny	winter

A static inductive learning algorithm (RULES-3 algorithm by Aksoy [3]) has been applied and the extracted rule set is given in Table 3.

Table 3. Initial rule base

Rule base
IF trees = yellow THEN season = autumn
IF temperature = moderate AND trees = bare THEN season = spring
IF temperature = high THEN season = summer
IF temperature = low THEN season = winter

Two thresholds for human-classified instance incorporation in existing rule base are chosen, *one* and *three* instances, respectively. Threshold = 1 means that classifier is rebuilt after every human classified instance. Let's examine how classifiers react to next three instances to classify.

A new instance "*trees = green AND temperature = moderate AND weather = rainy*" arrives. None of the existing rules fit the case. A human classifies it as "spring". In the example with threshold = 1 classifier is rebuilt and a new rule "*IF trees = green AND temperature = moderate THEN season = spring*" added. Another classifier includes full instance with its classification at the end of rules list (see Table 4).

Table 4. Rule base after one human classified instance

Threshold = 1	Threshold = 3
IF trees = yellow THEN season = autumn	IF trees = yellow THEN season = autumn
IF temperature = moderate AND trees = bare THEN season = spring	IF temperature = moderate AND trees = bare THEN season = spring
IF temperature = high THEN season = summer	IF temperature = high THEN season = summer
IF temperature = low THEN season = winter	IF temperature = low THEN season = winter
IF temperature = moderate AND trees = green THEN season = spring	IF temperature = moderate AND trees = green AND weather = rainy THEN season = spring

Now instance “trees = green AND temperature = moderate AND weather = cloudy” appears. First classifier covers it with the newly created rule. Second classifier transfers instance to a human who classifies it as “spring”. A new temporary rule is added (see Table 5).

Table 5. Rule base after two human classified instances

Threshold = 1	Threshold = 3
IF trees = yellow THEN season = autumn	IF trees = yellow THEN season = autumn
IF temperature = moderate AND trees = bare THEN season = spring	IF temperature = moderate AND trees = bare THEN season = spring
IF temperature = high THEN season= summer	IF temperature = high THEN season= summer
IF temperature = low THEN season = winter	IF temperature = low THEN season = winter
IF temperature = moderate AND trees = green THEN season = spring	IF temperature = moderate AND trees = green AND weather = rainy THEN season = spring
	IF temperature = moderate AND trees = green AND weather = cloudy THEN season = spring

The third instance is “trees = bare AND temperature = moderate AND weather = cloudy”. None of classifiers can define the class label for this instance and a human decides it is winter. Classifier rebuild should be executed in both classifiers, because the threshold in the second classifier is reached as the third human classified instance arrived (see Table 6).

Table 6. Rule base after three human classified instances

Threshold = 1	Threshold = 3
IF trees = yellow THEN season = autumn	IF trees = yellow THEN season = autumn
IF temperature = moderate AND trees = bare THEN season = spring	IF temperature = moderate AND trees = bare THEN season = spring
IF temperature = high THEN season= summer	IF temperature = high THEN season= summer
IF temperature = low THEN season = winter	IF temperature = low THEN season = winter
IF temperature = moderate AND trees = green THEN season = spring	IF temperature = moderate AND trees = green THEN season = spring
IF temperature = very low THEN season = winter	IF temperature = very low THEN season = winter

Both classifiers now look the same after temporary rules have been removed and generalization done. This case shows that classifier with lower threshold disturbed human more rarely (threshold = 1 asked for human advice twice while classifier with threshold = 3 did it three times). Classifier with a lower threshold managed to generalize the previous user decision sooner and the next similar instance could be classified by rule, not by human. Although immediate instance incorporation gives better results regarding classifier generalization and is likely to disturb human more rarely than the threshold higher than one, it requires more computations. Classifier with the threshold higher than one has smaller ability to generalize before the classifier gets updated. Yet a temporary rule helps in situations where exactly the same instance as the previous has to be classified and the system is expected to disturb human more rarely than a threshold based approach without any temporary rules.

5. Application Area

The proposed interactive inductive learning system has its characteristics that derive from the nature of the system. These characteristics also define the most appropriate problem areas for the particular approach.

5.1. Study Course Compatibility Analysis

One of the promising application areas of the proposed system is study course compatibility analysis in curriculum management. Because of globalization and student mobility there is a need for study course and study subject comparison to analyse their compatibility. Today curriculum development, maintenance and management require continuous monitoring of the knowledge provision space where different education providers offer a wide spectrum of knowledge acquisition opportunities. This is especially important in rapidly developing areas, such as ICT and Informatics [24]. Curriculum developers have to take into consideration knowledge offer of other education providers in line with such issues as scientific developments, industrial needs, etc. One of the problems that arise in monitoring the knowledge provision space is the necessity to compare different study courses. Taking into consideration the number of different education institutions operating inside the global knowledge provision space, the task of compatibility analysis, if performed only manually, is time consuming and requires frequent involvement of experts. Therefore a system that can support study course compatibility analysis is needed.

The following expectations and limitations are set from system's users to the system which is to be designed for helping course comparison.

- A human needs a system that helps in study course comparison.
- Human experts agree to invest their efforts at the initial period of system's establishment and training to achieve a comparative system that fits their needs and later gives feedback.
- The comparison procedure should be transparent and decision making steps explainable for a human.

These are the basic requirements for the course comparison system and the following subsection expands on them.

5.2. Features of Problem Domain

From a machine learning point of view course comparison is a classification task with course classification into one or more predefined classes (courses). Some courses are chosen to be target courses and other courses are compared (classified) to them. A more detailed description of the course comparison system is given in Section 6, after describing the course comparison task in a more structured way. The following features define problem domain.

- **Understanding decision making steps is important for human.** Not all potential users of the proposed system trust machine learning techniques if they can't verify how the results (classifications) were achieved. This condition defines the use of decision tree or rule generating algorithms among all machine learning methods thanks their explanatory power.

- **Small initial learning base.** It is expected that there would not be enough learning examples from human classified courses, especially at the start of classifier training. Therefore human involvement in helping classifier to learn is inevitable, it can be either showing more training examples or helping to classify new instances.
- **Many classes with similar probability to appear.** As study programmes usually consist of ten to fifty different study courses and there is no ground for preferring one course over the others, a default rule for assigning class to non-classifiable instances is not a proper approach. Along with a lack of examples to learn from, classifier with many classes is appropriate for an interactive approach in dealing with non-classifiable instances.
- **Multi-label class membership.** In the case of course classification a certain course can be similar to several other courses; therefore an assignment of more than one class is possible. This leads not only to the need for interactive learning approaches that can handle multi-label classification but also highlights the human role in classification because conceptual differences between classes are more vague than in strict single-label classification tasks.
- **Semi-structured data.** The study course is an issue that does not naturally possess well-defined attributes relevant for comparison of course contents. These attributes are also not always available and extractable from the information provided about the course. E.g., learning outcomes give distinctive information about courses, but they need to be mediated to one common framework to be automatically compared. Diverse and non-trivial structure of study course descriptions excludes the use of text classification approaches for solving the whole course comparison task.
- **Classification results are needed for a human, not only for a machine.** This need once more emphasizes the choice of explainable inductive learning techniques instead of, e.g. inductive logical programming, which is a powerful instrument, but lacks interpretability.

The above-mentioned features conform to the interactive inductive learning system's approach proposed in Section 3.

6. Interactive Inductive Learning System for Course Comparison

The study course is an issue that does not naturally possess well-defined attributes relevant for the comparison of course contents. While courses of a study programme may be viewed as knowledge classes where each course is relatively independent in terms of its title, running time and other attributes, in terms of course contents several study courses may overlap. Attributes used to describe study courses in the inductive learning system should not be only representative but also available. It is not always that education providers and trainers give detailed description of course contents [24]. However, learning outcomes usually are well described; therefore they can be used as a means for study course compatibility analysis. Besides learning outcomes other accessible attributes can be involved in classification, namely, study level, number of credit points for the course, etc. The comparison of learning outcomes have to be unified since verbal description of learning outcomes may vary for different educational institutions. Mapping from learning outcomes to a mediating competence

framework is created by a human expert. For this particular solution European e-Competence Framework is used. It is a European wide reference framework of ICT competences (see [25] for more details). So far the task of course mapping to particular competences in the framework is executed manually. This time-consuming work should be automated in the future. Some course attributes can be compared directly, namely, course level and number of credit points, while a mediating framework is needed to compare learning outcomes automatically.

The main stages of classifier formation and application for study course comparison are as follows.

- 1) There are courses that are considered as target courses (classes); other courses are compared to them. Attributes for those courses are extracted (study level, list of e-CF competences corresponding to a particular course, number of credit points). In the case of target courses, expert's classification is the same as the course title.
- 2) When all target courses are described and passed to classifier, a human expert should classify unfamiliar courses. The expert can decide that the unknown study course refers to more than one target course. The system receives course attributes and expert's classification and tries to find correspondences and infer rules from the provided examples.
- 3) After classifier has seen enough human classified courses to make judgements, the classification task of the new course (given as a set of attributes) can be passed to the system. If classifier is able to find an appropriate rule in its rule base a decision of course class (or classes) is made. If no rule can classify the course, a human is asked to classify it.

The proposed approach aims at minimizing human work in study course compatibility analysis. The results of initial experiments with the course comparison system confirmed that human involvement in the classification of new courses is useful when system can't make decision itself. It helps not only to assign the right class labels to a particular course but also improves classifier since new training examples (provided as human classifications) are very important, especially during the time the training set is small. The effect of the interactive approach is similar to that demonstrated by the example in Section 4.1.

So far the threshold based static learning approach with threshold = 1 has been used. The incremental learning approach is also available since multi-label classification with the aid of the problem transformation technique allows using any single-label classification method, including incremental algorithms [13].

7. Conclusions

The paper describes work in progress and complements previous research of the author [12, 20] on outlining the interactive inductive learning system. The proposed interactive inductive learning system deals with non-classifiable instances by asking a human for decision and improving its knowledge base with the rule derived from that human-made decision.

The main improvements since the earlier research are the following. Possible solutions of incorporation of human-made decision into rule base have been discussed. The process of creating and adding a rule to rule base following human classification

has been studied. Features of one application domains for applying the described system are described.

This paper proposes two models of human-made decision incorporation into rule base. One way is to use the threshold based static inductive learning method and relearn classifier from batch after one or more human classified instances arrive. If more than one instance is expected, previous instances are kept as rules until new classifier is built. Another way is to use the incremental learning method and incorporate human decision right into classifier as the example appears. It depends on particular problem domain, requirements, constraints, frequency of human classification, time needed for classifier rebuilding, learning algorithm, and other conditions which method to choose. Aspects of choosing between static or incremental learning algorithms have been analyzed.

The proposed interactive inductive learning system was used for automating course comparison in the area of curricula management. Initial experiments with the course comparison system are promising.

Acknowledgements

This work has been supported by the European Social Fund within the project „Support for the implementation of doctoral studies at Riga Technical University”.

References

- [1] N.W. Street, Oblique Multicategory Decision Trees Using Nonlinear Programming, *Inform Journal on Computing* **1** (2005), 25-31.
- [2] D.T. Pham, A.A. Afify, Applications of machine learning in manufacturing, *Intelligent Production Machines and Systems. 1st I*PROMS Virtual International Conference*, Elsevier, Great Britain (2005), 225-230.
- [3] M.S. Aksoy, Dynamic System Modelling Using Rules 3 Induction Algorithm, *Mathematical and Computational Applications* **1**, vol.10 (2005), 121-132.
- [4] K.J. Cios, L.A. Kurgan, Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms, *New Learning Paradigms in Soft Computing*, vol.84, Physica-Verlag GmbH, Heidelberg (2002), 276-321.
- [5] K. Kaufman, R.S. Michalski, The AQ18 Machine Learning and Data Mining System: an Implementation and User's Guide, Machine Learning and Inference Laboratory, George Mason University, USA (2000)
- [6] N. Ramakrishnan, The Pervasiveness of Data Mining and Machine Learning, *Data Mining for Software Engineering* (Aug. 2009), 28-29.
- [7] S. Chao, F. Wong, Y. Li, An Incremental and Interactive Decision Tree Learning Algorithm for a Practical Diagnostic Supporting Workbench, *Fourth International Conference on Networked Computing and Advanced Information Management*, vol.2 (2008), 202-207.
- [8] M.S. Aksoy, Applications of RULES-3 Induction System, *Proceedings of the Innovative Production Machines and Systems* (2008), 1-5.
- [9] P.E. Utgoff, Incremental induction of decision trees. *Machine Learning* **4** (1989), 161-186.
- [10] F. Ferrer-Troyano, J.S. Aguilar-Ruiz, J.C. Riquelme, Incremental Rule Learning based on Example Nearness from Numerical Data Streams, *Proceedings of the 2005 ACM Symposium on Applied Computing*, ACM, New York (2005), 568-572.
- [11] M.A. Maloof, S. Michalski, Incremental Learning with Partial Instance Memory, *Artificial Intelligence* **1-2**, vol.154 (2004), 95-126.
- [12] I. Birzniece, Interactive Inductive Learning System: The Proposal, *Proceedings of the Ninth International Baltic Conference*, Latvia, Riga (2010), 245-260.
- [13] F.A. Thabtah, P. Cowling, Y. Peng, Multiple labels associative classification, *Knowl. Inf. Syst.* **9**(1) (2006), 109-129.
- [14] G. Tsoumakas, I. Katakis, [Multi-Label Classification: An Overview](#), *International Journal of Data Warehousing and Mining* **3** (2007), 1-13.

- [15] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining Multi-label Data. *Data Mining and Knowledge Discovery Handbook* 2nd edition, O. Maimon, L. Rokach (Eds.), Springer, 2010.
- [16] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier Chains for Multi-label Classification, *ECML/PKDD*, Buntine et.al (Eds.), Vol. 5782/2009 (2009), 254-269.
- [17] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, *Machine Learning* **73** (2008), 185-214.
- [18] P. Clark, T. Niblett, The CN2 Induction Algorithm, *Machine Learning Journal* **3** (1989), 261-283.
- [19] J. Zhang, E. Nloedorn, L. Rosen, D. Venese, Learning Rules from Highly Unbalanced Data Sets, *Fourth IEEE International Conference on Data Mining* (2004), 571-574.
- [20] I. Birzniece, From Inductive Learning towards Interactive Inductive Learning, paper accepted for the *Scientific Proceedings of Riga Technical University, Applied Computer Systems*, Riga (2010), in press.
- [21] P. E. Utgoff, An Improved Algorithm for Incremental Induction of Decision Trees, Technical Report, University of Massachusetts (1994)
- [22] C. Giraud-Carrier, T. Martinez, An Incremental Learning Model for Commonsense Reasoning, *Proceedings of the Seventh International Symposium on Artificial Intelligence* (1994), 134-141.
- [23] R.O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification* 2nd Ed, Wiley-Interscience, 2000.
- [24] I. Birzniece, M. Kirikova, Interactive Inductive Learning Service for Indirect Analysis of Study Subject Compatibility, *Proceedings of the BeneLearn 2010*, Belgium, Leuven (2010), 1-6.
http://dtai.cs.kuleuven.be/events/Benelearn2010/submissions/benelearn2010_submission_9.pdf
- [25] European e-Competence Framework [Online]. Available: <http://www.ecompetences.eu/> [Accessed: Oct. 1, 2010].

Two Player Fair Division Problem with Uncertainty

Andre VESKI ^{a,1}, Leo VOHANDU ^{a,2}

^a *Ehitajate tee 5, Tallinn, Estonia*

Technical University of Tallinn, Institute of Informatics

Abstract. This paper analyses the territory fair division, problem initially posed by Hugo Steinhaus [1], by studying the solutions given by different algorithms on a large generated set of inputs for two players. Main algorithm used is Adjusted Winner, developed by S. Brams and A. Taylor [2]. We compare it to combinatorial enumeration and some algorithms proposed for experimentation by authors. Additionally we define measures to characterize the initial task and game theoretic measures to select the best solution. Moreover we extend the problem by allowing uncertainties in the players' value representation of items to be divided, based on the example of territorial division. For uncertainty management we use the belief system from Dempster-Shafer Theory [3].

Keywords. Fair division, Dempster-Shafer Theory

Introduction

Division problem is a more general task of the well known partitioning problem. Several different techniques have been developed to provide a fair division of goods or a single homogeneous good (e.g. cake). In general there are two classes of techniques: turn based information sharing and decision making; or providing full information to a mediator who then proposes a solution. In this paper we concentrate on the later type of algorithms, where we have all the information available from all players.

At first it may seem that a fair division is unreachable, because each player has their own subjective opinion about the goods being distributed. Ultimately just because of those subjective estimations, a fair division is possible. Unfortunately most of the world relies on experts' *objective* opinions and not on their own attitudes.

Initially this problem was described by Steinhaus [1], [4] as the problem of fair division. In his example two heirs have a territory to divide and both expect to get half of it. In Steinhaus' solution they both draw a vertical dividing line that would split the territory into two subjectively equal parts [1] (Figure 1). At first both players receive a piece of the territory they value more per area unit. In our example I_1 would be attributed to bidder B and I_3 to A . The remaining piece, I_2 , can be divided similarly or split randomly, since each of the players has already gained a half.

¹E-mail: andre.veski@gmail.com

²E-mail: leov@staff.ttu.ee

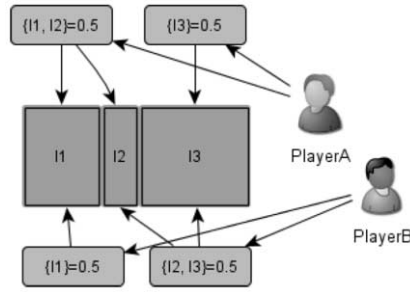


Figure 1. Division

To characterize fitness of a fair distribution, we use three measures proposed by Brams and Taylor [2]: *envy*, *equality* and *efficiency*. Envy shows how much a participant desires some one else's gains. Efficiency illustrates the final results of all players. A division is efficient when every participant got at least what he bargained for – with two player half of his initial values. Equality shows similarity in each participant's total gain. We will give a formal definition of these measures later in the paper. **In order for the division to be fair it has to be as envy-free, efficient and equal as possible.** There are cases where these measures contradict each-other and one can not drive all of them to the maximum. We also look how to fuse these three measures and use a single measure to evaluate the result such as Nash's Bargaining Solution.

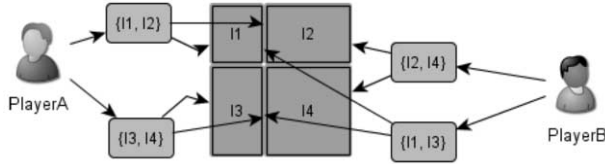


Figure 2. Conflicting division

In our paper we also add a level of generalization to the fair division problematique, based on an example of territory division. Instead of having non-crossing division lines as in Figure 1 we introduce crossing lines as on Figure 2. Hence we get a total of four items to be divided, but valuations only for item sets of two. This means we have some level of uncertainty. While we know how participants value part of their division, we don't know how it translates to each of the four individual sections as depicted on Figure 2.

1. More Formal Description of the Problem

We have a set S of n bidders S_1, \dots, S_n and a set I of m items I_1, \dots, I_m . Each bidder has a real-valued valuation function v_i that for each item $I_j \in I$ gives the value $v_i(I_j)$ that bidder S_i obtains if he receives I_j . A division of items among the bidders is a matching mutually exclusive subsets of items to bidders. For every bidder S_i, S_j where $i \neq j$ $D_i, D_j \subseteq I$ is an allocation such that $D_i \cap D_j = \emptyset$. For all bidders $\bigcup_{S_i \in S} D_i = I$. The total utility obtained is defined by $\sum_{S_i \in S} v_i(D_i)$.

As already mentioned, the problem of fair division is similar to the known NPC problem of partitioning, where items of different value need to be partitioned into n

Table 1. Valuations

v	I_1	I_2	I_3
v_A	0.44	0.6	0.50
v_B	0.50	0.5	0.45

distinct sets, all with equal total value. In fair division case, each bidder can be considered as a partition. And for each bidder $S_i \in S$ the value functions are the same, meaning that for any i, j and l the $v_i(I_l) = v_j(I_l)$. Often total equality between partitions is not achievable then we need to use a measure of fitness – equality between partitions. Having m items to be partitioned into two sets D_1 and D_2 , the goal is to minimize the difference

$$Equality(D_1, D_2) = \left| \sum_{I_i \in D_1} v_1(I_i) - \sum_{I_i \in D_2} v_2(I_i) \right| \quad (1)$$

For example having items with values $\{1, 2, 3\}$ then these can be easily divided into two sets of equal value $\{1, 2\}$ and $\{3\}$. The simplest algorithm to solve this task is Algorithm 1.1. Start from the largest element, assign it to a set, then second largest element to the set with a lower value, until there are no more items left. Partitioning problem can be generalized by adding some degrees of freedom. For example in fair division we have to take into account differences of opinion, i.e. the same item can have different values depending on the set they are assigned to – A or B . For simplicity, in the rest of the paper we will look at a problem with two bidders denoted $A = S_A$ or $B = S_B$.

Algorithm 1.1: MFP(I)

```

[Initialize]  $A, B \leftarrow \{\}$ 
while not empty  $I$ 
  do  $\left\{ \begin{array}{l} \text{[Find maximum] } \max\{v(I_1), \dots, v(I_m)\}, \text{ and } I_l \text{ is the maximal} \\ \text{[Select lower value set] if } Total(v(A)) \leq Total(v(B)) \\ \quad \text{then } A[\text{elements} + 1] \leftarrow I_l \\ \quad \text{else } B[\text{elements} + 1] \leftarrow I_l \end{array} \right.$ 
return  $(A, B)$ 

```

Let us take an example of fair division as presented on Figure 1. Assume that A and B have the value functions for items $\{I_1, I_2, I_3\}$ as in Table 1. The valuations are normalized so that for both A and B the total value would be 1. A good algorithm to solve this has been developed by Brams and Taylor [2], called Adjusted Winner (Algorithm 1.2), which has some similarities with Algorithm 1.1. In the first round the main goal is to return maximal efficiency by assigning each item to the highest bidder. In the second, adjustment step the result is equalized by giving most similarly valued items to the worst-off player. The algorithm tries to optimize all three criteria mentioned above.

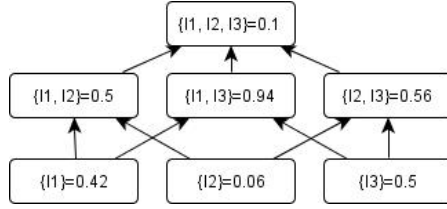


Figure 3. Valuations

Alternatively we may aim for highest utility as in Algorithm 1.3, which, as we shall see later, will in some cases yield a better result.

Algorithm 1.2: AWT(I)

```

[Initialize]  $A, B \leftarrow \{\}, m \leftarrow |I|, v \leftarrow 0$ 
for  $i \leftarrow 1$  to  $m$ 
  do  $\left\{ \begin{array}{l} \text{[Select largest and add]} \\ \text{if } v_A(I_i) > v_B(I_i) \\ \quad \text{then } A[\text{elements} + 1] \leftarrow I_i \\ \quad \text{else } B[\text{elements} + 1] \leftarrow I_i \end{array} \right.$ 
[Adjust A and B to be equal valued] if  $Total(v_A(A)) > Total(v_B(B))$ 
  then repeat
    for  $i \leftarrow 1$  to  $n$ 
      do  $\left\{ \begin{array}{l} c \leftarrow 2/(v_A(I_i)/v_B(I_i)) \\ \text{[Select best] if } v < c \\ \quad \text{then } v \leftarrow c, s \leftarrow i \end{array} \right.$ 
     $B[\text{elements} + 1] = I_s$ 
until  $Total(v_A(A)) \leq Total(v_B(B))$ 
  else if  $Total(v_A(A)) < Total(v_B(B))$ 
    then [Exchange]  $A \leftrightarrow B, \text{ancontinueatstep[Adjust]}$ 
return  $(A, B)$ 

```

Algorithm 1.3: MFT(I)

```

[Initialize]  $A, B \leftarrow \{\}, J \leftarrow \{I_i | i = \{1, \dots, m\}, |I_i| = 1\}, o \leftarrow |J|$ 
for  $i \leftarrow 1$  to  $o$ 
  do  $\left\{ \begin{array}{l} \text{[Select largest and add] if } v_A(J_i) > v_B(J_i) \\ \quad \text{then } A[\text{elements} + 1] \leftarrow J_i \\ \quad \text{else } B[\text{elements} + 1] \leftarrow J_i \end{array} \right.$ 
return  $(A, B)$ 

```

Let's look at the case where participant's valuations are uncertain, example on Figure 3. We have values for player A, with the value function $v_A(\{I_1, I_2\}) = v_A(I_1 \cup I_2) =$

0.5 and for individual items $v_A(\{I_1\}) = 0.42$ and $v_A(\{I_2\}) = 0.6$. The leftover value $v_A(I_1 \cap I_2) = v_A(\{I_1, I_2\}) - v_A(\{I_1\}) - v_A(\{I_2\}) = 0.5 - 0.42 - 0.06 = 0.02$ is the value that comes from owning both of these items together. In other words, it is the value of the connection between those two items. In Dempster-Shafer Theory (henceforth DST) [3] we have upper and lower bound values for all items. Going forward, we describe briefly the necessary part of DST [3]. On a valuation lattice (Figure 3) there are at least three ways of looking at each item set:

1. Total value of items which an item contains - belief
2. Total value of items in which an item is part of - plausibility
3. Total value of items in which an item is not part of and does not contain itself - doubt

The base values for each item or set of items is called mass or basic probability assignment [3] and is the value only for that particular item. The mass values for all item sets in the power set add up to 1 and value of an empty set is 0. Meaning that for single items the mass is the value of that item and in larger item sets the value of the connection (intersection). In our example (Figure 3) the mass $m_A(I_1, I_2) = v_A(I_1 \cup I_2) = 0.02$ and the belief value $b_A(I_1, I_2) = v_A(I_1, I_2) = 0.5$. The belief function from DST corresponds to our idea of a value function so that $b_A(I_i) = v_A(I_i)$. Here $\sum_{I_i \in I} m_A(I_i) = 1$, whereas $\sum_{I_i \in I} v_A(I_i) \geq 1$. The definitions for item set value functions are given below.

Definition 1. Belief is a sum of masses from all the sets where observed element I_i is a part of or in other words a lower bound for an item set A that contains the certain knowledge about the item set. On Figure 4 we have belief for $I_6 = \{I_2, I_3\}$ presented with gray background.

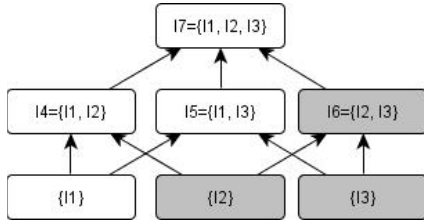
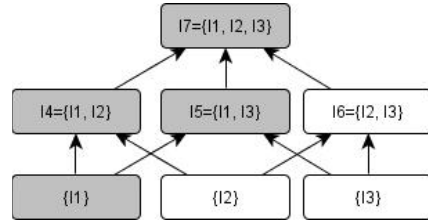
$$Belief(I_i) = b(I_i) = \sum_{I_j \subset I_i} v(I_j) \quad (2)$$

Definition 2. Plausibility is a sum of masses from all the item sets where the union with the observed element I_i is not an empty set or in other words an upper bound for a set I_i that I_i would have if it got assigned all the uncertain values. On Figure 5 there is plausibility for I_1 with a gray background.

$$Plausibility(I_i) = p(I_i) = \sum_{I_j \cap I_i \neq \emptyset} v(I_j) \quad (3)$$

Definition 3. Doubt is a sum of masses from all the item sets where the union with the observer element I_i is an empty set or value that I_i can never have even if it got assigned all the uncertain values. On Figure 4 there is doubt for I_1 presented with a gray background.

$$Doubt(I_i) = d(I_i) = \sum_{I_j \cap I_i = \emptyset} v(I_j) \quad (4)$$

Figure 4. Belief I_6 , Doubt I_1 Figure 5. Plausibility I_1

2. Measuring Solutions and Tasks

With each additional degree of freedom there are more possibilities to evaluate the fitness of a solution. On the simple partitioning task there is a measure of equality between sets. As the number of partitions grows, new measures come up such as a total difference of values in pairwise comparison etc.

A fair division is usually described by three measures according to Brams and Taylor [2]: efficiency, envy and equality and we also use total product. The latter is also used by Nguyen and Kreinovich [5] and is known in the game theory as Nash's Bargaining Solution [6].

Definition 4. (Pareto) Efficiency is the total value of the solution. This is actually a condition where no player can be made better off by making someone else worse off. But if an item can change owners and create more value from that, the initial owner can be compensated by some uniform measure – e.g. money.

$$Efficiency(A, B) = ef(A, B) = \sum_{I_i \in A} m_A(I_i) + \sum_{I_i \in B} m_B(I_i) \quad (5)$$

Definition 5. Envy is the amount by which in one player's valuations other players result was larger than his. The total envy is the total sum on pairwise comparisons and consists of two parts:

$$Envy(A, B) = en(A, B) = \max \left(\left(\sum_{I_i \in B} m_A(I_i) - \sum_{I_i \in A} m_A(I_i) \right), 0 \right) \quad (6)$$

$$Envy(B, A) = en(B, A) = \max \left(\left(\sum_{I_i \in A} m_B(I_i) - \sum_{I_i \in B} m_B(I_i) \right), 0 \right) \quad (7)$$

$$TotalEnvy(A, B) = Envy(A, B) + Envy(B, A) \quad (8)$$

Definition 6. Equality is the amount by which end results differ for each player and is calculated as a sum of pairwise differences

$$Equality(A, B) = eq(A, B) = \left| \sum_{I_i \in A} m_A(I_i) - \sum_{I_i \in B} m_B(I_i) \right| \quad (9)$$

Definition 7. Product is the total product of all players end results

$$Product(A, B) = pr(A, B) = \sum_{I_i \in A} m_A(I_i) \cdot \sum_{I_i \in B} m_B(I_i) \quad (10)$$

Envy and equality are somewhat similar. In partitioning task we have equality as our measure of fitness. Since each subset of items has now different values based on every players individual preferences, we also need the envy measure to assess fitness from all players viewpoints.

Efficiency is a new kind of measure. With simple partitioning this measure is equal to the total value of items and is always the same, regardless of the actual solution. In fair division, since the total values of partitioned subsets are different in each solution, it is important to make sure that we would get the maximum possible total value.

Similarly measuring the goodness of a solution, there are measures to characterize the initial players valuations. We will use these to generalize the input in order to create a relationship with the output.

Definition 8. Conflict is defined as a conflict measure in the DST [3]

$$Conflict(A, B) = c(A, B) = 1 - \sum_{i=1}^n \sum_{j=i, I_i \cap I_j = \emptyset}^n m_A(I_i) \cdot m_B(I_j) \quad (11)$$

Definition 9. Difference is a pairwise difference in participants' valuations. This is recommended by authors and as we see later it has a good correlation with the solution.

$$Difference(A, B) = d(A, B) = \frac{\sum_{i=1}^n |v_A(I_i) - v_B(I_i)|}{2} \quad (12)$$

Difference describes the total difference in players' value functions. Looking at the Algorithm 1.2, the more different the players' value functions are, the more efficient the solution should be. In extreme cases where valuations are completely opposite, the resulting gain would be double of that the players initially subjectively expected.

Definition 10. Uncertainty is an amount of uncertainty in the valuations as total sum of mass on item sets with greater volume than 2.

$$Uncertainty() = un() = \sum_{i=1, |I_i| > 1}^n m_A(I_i) + m_B(I_i) \quad (13)$$

In Table 2 we have presented an example calculation of definitions given above. It is based on problem from Table 1 where the result is for $A = \{I_1\}$ and for $B = \{I_2, I_3\}$.

Table 2. Example calculations

	Player A	Player B
Result	0.50, 0.6	0.50
Efficiency	$0.6 + 0.50 + 0.50 = 1.06$	
Envy	$0.67 - (0.27 + 0.6) = 0.34$	$0.61 + 0.5 - 0.34 = 0.33$
Equality	$0.67 - 0.61 - 0.05 = 0.01$	
Product	$0.56 \cdot 0.50 = 0.28$	
Conflict	$1 - 0.44 \cdot 0.05 - 0.44 \cdot 0.45 - 0.06 \cdot 0.5 - 0.06 \cdot 0.45 - 0.5 \cdot 0.5 - 0.5 \cdot 0.05 = 0.448$	
Difference	$\frac{1}{2} \cdot (0.44 - 0.50 + 0.6 - 0.5 + 0.50 - 0.45) = 0.06$	
Uncertainty	0.0	

3. Algorithms Modifications for Uncertainty

So far we have examined two algorithms for fair division: Maximal First and Adjusted Winner for two bidders. As a next step we need to figure out how to handle uncertain valuations in these algorithms. In authors' views handling uncertainty could be done in at least two ways.

- 1. Using different value functions from DST to determine a value for item comparison, e.g. belief or plausibility
- 2. Start the division process with different volumes sets of items, e.g. on the second level of the valuation lattice where all item sets have a cardinality of two. Without uncertainty, all item sets have a cardinality of one.

In the paper we will explore only the first. More precisely we shall compare results from three algorithms.

- 1. Enumerating all possible combinations (Algorithm CT and Algorithm C)
- 2. 2-step Adjusted Winner (Algorithm AWT and Algorithm AW)
- 3. Maximal valued First (Algorithm MFT and Algorithm MFL)

In other words, all algorithms will be used in two models.

- 1. Model with uncertainty (Algorithms C, AW and MFL)
- 2. Model without uncertainty (Algorithms CT, AWT and MFT)

Algorithms AW (Algorithm 3.1) and MFL (Algorithm 3.2) are modifications of their certain world counterparts AWT (Algorithm 1.2) and MFT (Algorithm 1.3). With AW we still loop only over single items, but when comparing the item sets we use the plausibility function. We also add a step to check if we can add some item sets already covered by single items. In generalizing MFT to MFL we use the belief measure, not just mass, but again we loop through single items and add larger item sets at the final stage.

Algorithm 3.1: AW(I)

```

[Initialize]  $A, B \leftarrow \cdot, J \leftarrow \{I_i | i = \{1, \dots, m\}, |I_i| = 1\}, o \leftarrow |J|$ 
for  $i \leftarrow 1$  to  $o$ 
  do  $\left\{ \begin{array}{l} \text{[Select largest and add] if } p_A(J_i) > p_B(J_i) \\ \quad \text{then } A[\text{elements} + 1] \leftarrow J_i \\ \quad \text{else } B[\text{elements} + 1] \leftarrow J_i \end{array} \right.$ 
[Adjust]  $v \leftarrow 0$ 
if  $Total(v_A(A)) > Total(v_B(B))$ 
  then repeat
    for  $i \leftarrow 1$  to  $o$ 
      do  $\left\{ \begin{array}{l} a \leftarrow b_A(J_i) b \leftarrow b_B(J_i) c \leftarrow 2/(a/b) \\ \text{[Keep best matching item] if } v < c \\ \quad \text{then } v \leftarrow c, s \leftarrow i \end{array} \right.$ 
       $A[i] \leftarrow nil, B[i] = J_i$ 
    until  $Total(v_A(A)) \leq Total(v_B(B))$ 
    else if  $Total(v_A(A)) < Total(v_B(B))$ 
      then [Exchange]  $A \leftrightarrow B$ , and continue at step[Adjust]
 $K \leftarrow \{I_i | i = \{1, \dots, m\}, |I_i| > 1\}, q \leftarrow |J|$ 
for  $i \leftarrow 1$  to  $q$ 
  do  $\left\{ \begin{array}{l} \text{if } K_i \subset A \\ \quad \text{then } A[\text{elements} + 1] \leftarrow K_i \\ \quad \text{else if } K_i \subset B \\ \quad \text{then } B[\text{elements} + 1] \leftarrow K_i \end{array} \right.$ 
return  $(A, B)$ 

```

Algorithm 3.2: MFL(I)

```

[Initialize]  $A, B \leftarrow \cdot, J \leftarrow \{I_i | i = \{1, \dots, m\}, |I_i| = 1\}, o \leftarrow |J|$ 
for  $i \leftarrow 1$  to  $o$ 
  do  $\left\{ \begin{array}{l} \text{[Select largest and add] if } b_A(J_i) > b_B(J_i) \\ \quad \text{then } A[\text{elements} + 1] \leftarrow I_i \\ \quad \text{else } B[\text{elements} + 1] \leftarrow I_i \end{array} \right.$ 
 $K \leftarrow \{I_i | i = \{1, \dots, m\}, |I_i| > 1\}, q \leftarrow |J|$ 
for  $i \leftarrow 1$  to  $q$ 
  do  $\left\{ \begin{array}{l} \text{if } K_i \subset A \\ \quad \text{then } A[\text{elements} + 1] \leftarrow K_i \\ \quad \text{else if } K_i \subset B \\ \quad \text{then } B[\text{elements} + 1] \leftarrow K_i \end{array} \right.$ 
return  $(A, B)$ 

```

Table 3. Valuation example

	I_1	I_2	I_3	I_4	I_1, I_2	I_3, I_4	I_1, I_3	I_2, I_4
Player A	0	0	0	0	0.5	0.5	0	0
Player B	0	0	0	0	0	0	0.5	0.5

Algorithm 3.3: C/CT(I)

```
[Initialize]  $l \leftarrow 0, A, B \leftarrow \{\}$ 
repeat
   $A \leftarrow \text{Combination of Items}(I)$ 
   $B \leftarrow I - A$ 
  [Compare the new solution with the current solution] if  $pr(A, B) > l$ 
    then  $l \leftarrow pr(A, B)$ 
[All combinations tested]
until  $\text{OutOfCombinations}()$ 
return  $(A, B)$ 
```

4. Algorithm Comparison Setup

To compare the results for the algorithms we look at a large set of generated inputs, basically taking assumptions for both player on their values for single items (I_1, I_2, I_3 and I_4). Our goal is to look at many different settings. Some of them have uncertainty, some of them are closely valued and some are very different. To make sure we have all the possible settings we generate a full set of combinations. Since $v_A(\{I_1, I_2\}) = v_A(\{I_3, I_4\}) = 0.5$ will hold and respectively $v_B(\{I_1, I_3\}) = v_B(\{I_2, I_4\}) = 0.5$. If we know that $v_A(I_1) = 0.1$ then we also know that $v_A(I_2)$ can be at most 0.4). Meaning that for the valued item set $H = \{H_1, H_2\}$ we have $6 + 5 + 4 + 3 + 2 + 1 = 21$ ways to share the $v_A(H) = 0.5$ between the individual items H_1 and H_2 . Since there are two such item sets for both players, we get $21^{2*2} = 194481$ examples. We don't need examine all the examples. To reduce the amount of computation we skip mirrored valuations, such that $v_{B_2}(I_1) = v_{A_1}(I_1), v_{B_2}(I_3) = v_{A_1}(I_2), v_{B_2}(I_2) = v_{A_1}(I_3), v_{B_2}(I_4) = v_{A_1}(I_4), v_{A_2}(I_1) = v_{B_1}(I_1), v_{A_2}(I_2) = v_{B_1}(I_3), v_{A_2}(I_3) = v_{B_1}(I_2)$ and $v_{A_2}(I_4) = v_{B_1}(I_4)$. Assuming that Player A is A_1 and A_2 from example 1 and example 2 respectively and B_1 and B_2 Player B. This leaves us with 97461 examples.

- 1. Table 3 example has high uncertainty and high conflict.
- 2. Table 4 example has no uncertainty, has high conflict and high difference.
- 3. Table 5 has some conflict, some uncertainty and by our definition no difference.

Additionally to compare algorithms that can't handle uncertainty, we need to transform the input. For this we use the plausibility measure. For all single items we calculate their plausibility and then normalize that to add up to 1. This calculation basically distributes the uncertainty uniformly between the items.

Table 4. Valuation example

	I_1	I_2	I_3	I_4	$I_{1,2}$	$I_{3,4}$	$I_{1,3}$	$I_{2,4}$
Player A	0.5	0	0.5	0	0	0	0	0
Player B	0	0.5	0	0.5	0	0	0	0

Table 5. Valuation example

	I_1	I_2	I_3	I_4	$I_{1,2}$	$I_{3,4}$	$I_{1,3}$	$I_{2,4}$
Player A	0.2	0.2	0.2	0.2	0.1	0.1	0	0
Player B	0.2	0.2	0.2	0.2	0	0	0.1	0.1

Table 6. Correlations

	Efficiency	Envy	Inequality	Product
Efficiency	1			
Envy	-0.21	1		
Inequality	0.18	0.80	1	
Product	0.65	-0.76	-0.59	1

5. Assessing Metrics

Based on the metrics presented, which would be the best to maximize? We have presented Envy, Efficiency, Equality and Nash’s Bargaining Solution. In Brams and Taylor [2] Adjusted Winner algorithm efficiency is preferred to reach at least 1 and after that the aim is to equalize the solution. In general we wish to go in all directions at once: minimize envy, maximize efficiency and equality. It appears that Nash’s Bargaining Solution does just that.

On Figure 6 we can see the relation between all the four metrics. These values have been generated from all possible solutions (16) to our set of 97461 examples using Algorithm 3.3. This results in total about 1.6 million cases. In a rough summary the relationships between the metrics are (based on Figure 6):

- 1. Product has a good correlation with Efficiency (Table 6)
- 2. The highest Product is almost always with low Envy. There are always tasks where Envy is unavoidable, the extreme case being the example in Table 3 in the previous section
- 3. With a higher inequality the product approaches 0, as can be seen from the chart (Figure 6) starting from bottom left and moving to upper right

Since all the correlations with the product are in the right direction (Table 6) as stated: maximizing efficiency and equality and minimizing envy and all correlation coefficients with Nash’s Bargaining Solution are above 0.5 which makes it a good metric to assess the fitness of our solutions. Moreover there is more justification on using the Nash’s Bargaining solution in [6] and [7].

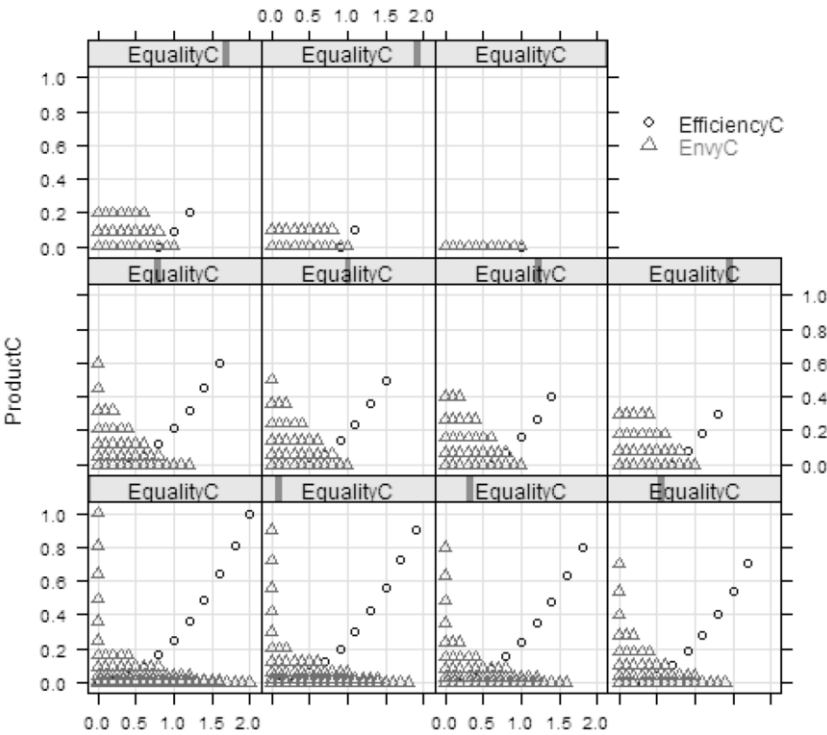


Figure 6. Solution metrics

6. Initial Results

Solving tasks in the model without uncertainty, the resulting solution is usually well predicted by the initial difference of opinions between the players. The greater the difference the better the result will be as on Figure 7.

As expected, adding uncertainty creates additional level of complexity. The differences of opinion don't have such strong impact on the result as before. On Figure 8 we see that the difference itself does not give as good an explanation as before. Difference still has a similar direction as in the model without uncertainty, the best result is achieved only with the biggest difference, but not the other way around i.e. larger difference does not guarantee a better solution. Moreover from Figure 8 we see that there is no definite impact by uncertainty and conflict either. Although there is visible direction with uncertainty, i.e. the greater uncertainty implies lower product value, but with decreasing uncertainty the variability of the Nash's Bargaining Solution increases.

Next let us compare results from different algorithms, with (Figure 10) and without (Figure 9) uncertainty. On the certain world (Figure 9) we see that all the algorithms produce quite similar results. Most of the results from algorithms AWT and MFT produce results quite near to the ideal, both have some deviation from CT, but not very much. In general there is room for improvement for both algorithms, although AWT manages always to beat MFT.

Adding uncertainty to algorithms the best algorithm is not that obvious anymore. When comparing algorithms AW and MFL it is visible on Figure 10 that MFL is in some

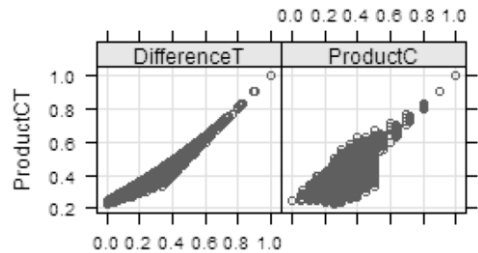


Figure 7. Difference without Uncertainty

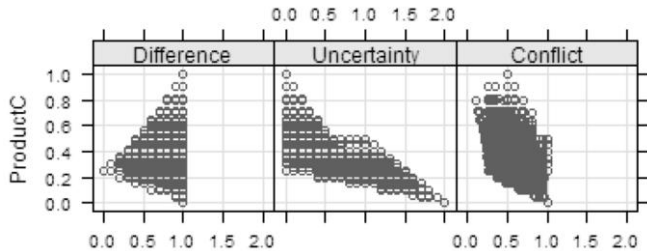


Figure 8. Difference with Uncertainty

cases able to achieve better results than AW. This means that second stage in Algorithm AW, adjustment for equality, has some negative impact on other metrics in terms of Nash’s Bargaining Solution. This is definitely one of the places to start creating an even better algorithm.

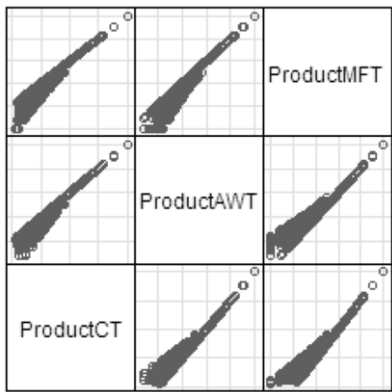


Figure 9. Results without Uncertainty

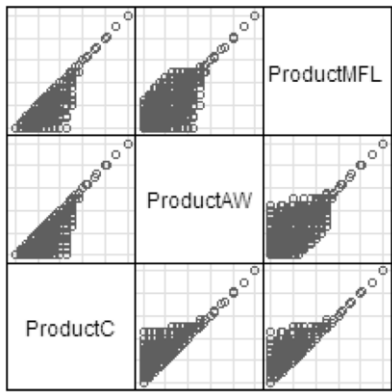


Figure 10. Results with Uncertainty

7. Conclusion

It has been confirmed once more that Nash’s Bargaining Solution contains all the necessary properties of the bargaining game. While maximizing utility value it also tries to balance mutual equality and minimize envy. It is also important to understand the limita-

tion we have here; we just have looked at a limited, but a large set of examples. It might be that examples included here belong to a separate class of behavior.

Looking at the various charts it is apparent that the quality of the solution is determined by initial limits we have set on item values as seen on Figure 10. Having certain valuation it is clear that the outcome is defined by the initial differences in players valuations. Although in the case of uncertainty the determination is not that clear. But in general, as would be expected by intuition, adding uncertainty also degrades the possible solution. Uncertainties impact on the solutions for more complex problems (three and more players) remains to be investigated. What would be an efficient algorithm in that case?

Moreover the setup of our test environment is quite limited. Currently we have only tested the case of two participants, who had four items to share. Making the task more complex can reveal new insights on the algorithms. Next steps on the test settings should be:

1. Expanding the task for 3 and more participants
2. Expanding the item space and therefore the valuation lattice will be more complex
3. Extending the initial task beyond the territory division to allow more freedom, i.e. item set values between $[0;1]$

References

- [1] Hugo Steinhaus, The problem of fair division, *Econometrica*, 16, 101-104 (1948)
- [2] Steven J. Brams, Alan D. Taylor. *Fair Division: From cake-cutting to dispute resolution*, Melbourne (1996)
- [3] William L. Oberkampf, Jon C. Helton. *Evidence Theory for Engineering Applications Appeared in Engineering Design Reliability Handbook*, Danvers (2004)
- [4] Hugo Steinhaus. *Mathematical Snapshots*, New York (1969)
- [5] Hung T. Nguyen, Vladik Kreinovich, How to divide a Territory: A New Simple Formalism for Optimization of Set Functions, *International Journal of Intelligent Systems*, 223-251 (1999)
- [6] Duncan Luce, Howard Raiffa. *Games and Decisions: Introduction and critical survey*, Dover (1989)
- [7] Ken Binmore. *Natural Justice*, Oxford (2005)

Visualization of Airport Procedures in Time Critical Decision Support Systems

Kristina LAPIN, Vytautas ČYRAS and Laura SAVIČIENĖ

Faculty of Mathematics and Informatics, Vilnius University, Lithuania

Abstract. This paper reports on the development of two alternative visualizations of airport requirements for approach and departure trajectories. The work is undertaken within the FP6 SKY-Scanner project that develops a new lidar (laser radar, Light Detection And Ranging) equipment. Radar and lidar data fusion provides the SKY-Scanner system with exact aircraft position within the aerodrome traffic zone (ATZ). We propose a new surveillance paradigm which enables the controller to estimate visually a current situation without mental calculations of altitude and distance. Two alternative prototypes are based on innovative ideas proposed in aeronautics. A demonstration of alternative designs supports the discussion, invites users to compare solutions and thus provides developers with informative user feedback. The paper proposes visualizations based on 2D and 3D view integration and on pure 3D photorealistic view. The paper explores how innovative visualizations create new opportunities for controlling air traffic in the ATZ.

Keywords. Controllers' needs, situational awareness, air traffic control, departure and approach procedures, 2D and 3D visualization

Introduction

The SKY-Scanner system is a novel laser technology that aims at detecting and tracking aircraft up to at least 6 nautical miles from the aerodrome traffic zone (ATZ) barycentre [1]. The work is conducted within the FP6 SKY-Scanner¹ project. The presented work is performed as part of the Single European Sky Air traffic management Research (SESAR) programme [2]. SESAR aims at increasing the capacity and efficiency of the European Air Traffic Management (ATM) system [3]. SKY-Scanner explores, among others, avoiding the risks identified by SESAR.

Air traffic control (ATC) is a service provided by the ground-based controllers for the purpose of preventing collisions and maintaining an orderly flow of traffic [4]. Each airport determines its landing and take-off procedures (airport procedures, for short) that air controller assigns for the aircraft. Each landing and take-off is executed according to the assigned procedure. A SKY-Scanner data model which is derived from an airport procedure is provided in [5].

ATM systems have long implementation cycles. Research and development projects generate ideas that are elaborated in subsequent projects. A proposed solution is verified from different perspectives. NASA Technology Readiness Levels (TRLs)

¹ EU FP6 TP1.4 Aeronautics and space, TREN-4-Aero. Title: "Development of an Innovative LIDAR Technology for New Generation ATM Paradigms" (SKY-Scanner), 2007-2010, <http://www.sky-scanner.it/>

form a systematic metric/measurement system that supports maturity assessments of a particular technology and a consistent comparison of maturity between different types of technology [6]. The lowest first level (TRL-1) addresses basic principles observed and reported. In the second level technology, the concept and/or application is formulated. Our research further develops TRL-1 and TRL-2 level ideas of 3D-in-2D Planar Displays for ATC project [7]. Proposed novel ideas of representing relevant 3D information within 2D planar display were elaborated to TRL-3. Second prototype implements 3D photorealistic visualization. Our studies set the proposed ideas into a practical context defined by SKY-Scanner project requirements. Prototypes are developed for laboratory-based studies in order to validate physically that the analytical predictions are correct.

The research addresses a decision support system (DSS) for a new generation air traffic management paradigm (Figure 1). The topics include aircraft collision risk evaluation and trajectory adherence to airport procedures.

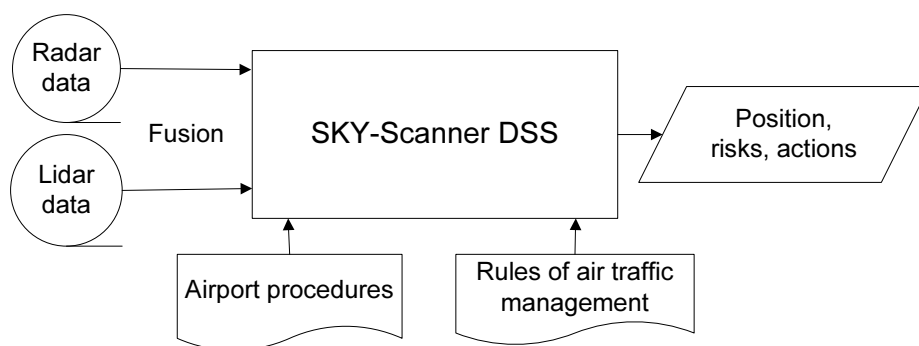


Figure 1. The scope of SKY-Scanner

The SKY-Scanner system serves for the surveillance of aircraft landing and take-off in the terminal area. The manoeuvres are performed at a low altitude where radar equipment functions below a needed accuracy. Therefore current systems leave the final landing and initial take-off phases under pilot's responsibility.

Lidar measurements are exact when directed to the target. An approximate position received from the radar facilitates lidar's deflection to the target. When the target is found, the lidar switches to the tracking mode and provides the exact target position for the SKY-Scanner system.

SKY-Scanner DSS is aimed at extending controller's capabilities to control the adherence of aircraft landing and take-off trajectories with airport procedures. The controller gives the pilot a clearance to land. The pilot performs the maneuver and reports. The controller can also observe the landing visually from the tower. The SKY-Scanner system visualizes an actual situation on the screen. The system enables a human operator to estimate visually whether the aircraft meets trajectory constraints.

This also enables tracking the flight from the beginning till the end. Allowed distances between aircraft are extracted from air traffic management rules. Airport procedures define trajectories that must be followed by the crew. DSS output provides the fused aircraft position and estimated violation risks. Lidar and radar data are fused in order to better estimate the risks and to propose corrective actions for the controller.

We start with an analysis of time-critical decision support models. The reason is that ATM activities need a real time response from the air traffic controller. The

analysis highlighted that appropriate visualization is the most important aspect to support time critical decision.

Then we select innovative visualizations for ATM from previous research projects for which extensive user research was performed and user feedback was asserted as positive. The most promising from this perspective were visualisation ideas of 3D-in-2D Planar Displays project. Despite the primary goal of SKY-Scanner being a novel technology, the developed DSS should be validated.

Informal feedback from experienced users is a valuable source of information. Experienced interaction designers recommend that end users should be provided with alternative solutions [8]. By offering a variety of visual designs, a designer invites the end user to compare alternatives. The user expresses what is perceived as advantages and drawbacks instead of unproductive comments such as “like this” or “hate that”. For this reason a photorealistic 3D view was created as an alternative design.

The report on accidents in 1980-2001 concludes that 12.7% accidents occurred in take-off and landing phases [9]. The analysts of the ATM-related accidents arrived at the conclusion that the causal factors were low visibility and incorrect or inadequate instruction/advice given by ATC. A frequent event factor in landing accidents was non-adherence to procedures by flight crew. Thus the current situational awareness is insufficient.

A standard 2D radar display does not show the third dimension. The controller has to interpret alphanumerical characters and to hold the altitude in his mind [7]. Current systems present the recent situation in 2D plan view where aircraft horizontal position is shown. The third dimension (altitude) and speed are presented with the aircraft label. In order to follow the actual situation and to indicate possible future troubles, controllers perform mental calculations of altitude and speed. Altitude visualization enables the controller to reduce mental overload and to improve situational awareness. These goals accord with the ATM vision of SESAR [2].

The next section describes decision making processes from human perspective. From the modeling perspective, departure is analogous to approach. In section 2, we deal with SKY-Scanner decision support model. Section 3 presents contemporary visualization approaches for ATC. Section 4 describes the SKY-Scanner visualization alternatives. Finally conclusions are drawn.

1. DSS Models for Time Critical Systems

Decision making is the process of selecting a choice or course of action from a set of alternatives. The human cognitive processes underlie most of decision making models. In this section we analyze human cognitive processes and time critical decision-making models in order to define a paradigm for the SKY-Scanner DSS.

Human decision making processes are facilitated using a variety of reasoning techniques. One of them is analogical reasoning where novel solutions are inferred via analogy with known solutions and methods. Analogical reasoning includes the following serial procedures [10]:

1. Encoding: translating stimuli to internal (mental) representations.
2. Inference: determining the relationship between problems.
3. Mapping: determining correspondences between new and old items.
4. Application: execution of the decision process.
5. Response: indicating the outcome of the reasoning process.

Since the steps in this reasoning process proceed in a serial manner, temporal ordering and timing of decision support is critical to improving time-critical decision-making. Regardless of the stimuli, the encoding step is the largest single component of the reasoning process, taking about 45% of the overall reasoning time [11]. For example, the encoding of words takes longer than the encoding of schematic pictures, implying that reducing text in displays will facilitate faster decision-making. Thus, in general, time critical decision making displays should concentrate on facilitating quicker encoding, possibly by more intuitive visualization.

Time critical decision making models are studied mainly in military context. Most models describe the human process of decision making as serial staged processes that include steps centered on information gathering, likelihood estimation, deliberation, and decision selection [11].

There are two philosophical approaches toward decision-making: the rational (or logical, or analytical) approach vs. the naturalistic (or action-based, or recognition-primed) approach.

The rational decision-making model assumes that a clear set of alternative choices can be generated in advance and their likely outcomes predicted with a significant degree of confidence. This model presumes to be objective, by establishing criteria, weighting them, and then choosing the highest utility.

The action-based or naturalistic model based upon imposing an interpretation of an ambiguous situation [12, 13]. This model assumes that knowledge results from actions and observing consequences. There is an inherent assumption that taking a decision too much information can be detrimental. The naturalistic model assumes that it is not feasible to fully quantify a situation and find a solution mathematically. A human decision maker makes a decision based on observed subject actions.

Summarizing, the rational model is objective but requires calculating the utility of each alternative, whereas the naturalistic model highlights the need to provide the human with relevant information. Avoiding unnecessary details facilitates stimuli encoding process.

2. Decision Support for ATM

Not all decision-making models can be applied for any situation. Many models share common aspects and attributes but differ in the order, area of emphasis or underlying assumptions. A significant aspect is limited or ample time to analyze the situation before making a decision [11]. Simple decisions are taken during landing or take-off, for example, turn left or right. There is no space for trial and error. Optimization level is important as corrections are not possible.

Landing procedures are based on strict rules. Presenting the aircraft actual position with respect to a visualized landing/take-off procedure enables controllers to visually follow the situation and to estimate whether aircraft adheres the assigned procedure. Terrain peculiarities adjust airport procedures to the natural context (Figure 2).

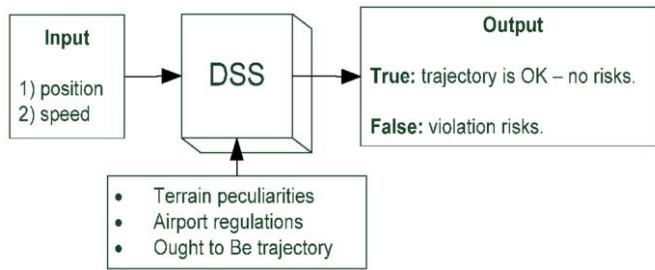


Figure 2. SKY-Scanner decision support

Our analysis of time critical decision-making models revealed the key features for the DSS being developed. First, from the human cognitive perspective, quick encoding is a key factor that facilitates human decision-making. This is achieved by providing an intuitive visualization. Since pictures are encoded faster than symbols, a requirement is to present information graphically. Second, naturalistic decision-making models in the context of tracking aircraft require presenting the airport procedures intuitively. From the modeling perspective, the procedures define a relation between aircraft track, distance from the runway threshold and altitude. Thus, a proper visualization of this relation is needed.

3. Related Work on DSS Visualization

A 2D radar display combines graphical and symbolic information. The geographical aircraft position is shown on 2D plan while the altitude is presented by symbols. The novel 3D visualizations enable presenting the altitude as the third dimension and reducing the amount of symbolic information. A 3D view requires significantly less cognitive effort to interpret altitude information. It supports more informed decision-making on the vertical dimension. However, it is easy to clutter 3D view with unimportant details aiming to render as realistic picture as a view through the controller's window. 3D interfaces should be minimalistic and abstract despite the temptation to provide beautiful realistic landscapes.

According to the SKY-Scanner requirements, DSS output has to be implemented on traditional 2D displays. 3D visualizations in aeronautics and geographical domains address the following sources:

- Space-time cube representations where two planar dimensions represent geographical space and the third vertical spatial dimension is time [14]
- Strict 3D visualization of air traffic concepts developed for free flight in Hughes Research Laboratories [15]
- Visualizations proposed in the project named "3D-in-2D Planar Displays for ATC" [16].

3.1. Space-Time Cube Visualizations

Space-time cube (STC) is a structure that is used to depict target activities in space-time context. STC adopts a three-dimensional orthogonal viewpoint [17]. The horizontal axes record the position and location changes of objects. The vertical axis

provides an ordered and synchronized sequence of events. In its basic appearance these images consist of a cube with geography on its base (along the x- and y-axis), while the cube's height represents time (z-axis).

A typical STC contains object trajectories, also known as space time-paths, of an object moving in time. Figure 3 shows an example of human space-time behavior. According to this technique, points in three-dimensional space, where the vertical dimension corresponds to time, represent positions of an object at different time moments. Lines connect the points corresponding to consecutive moments.

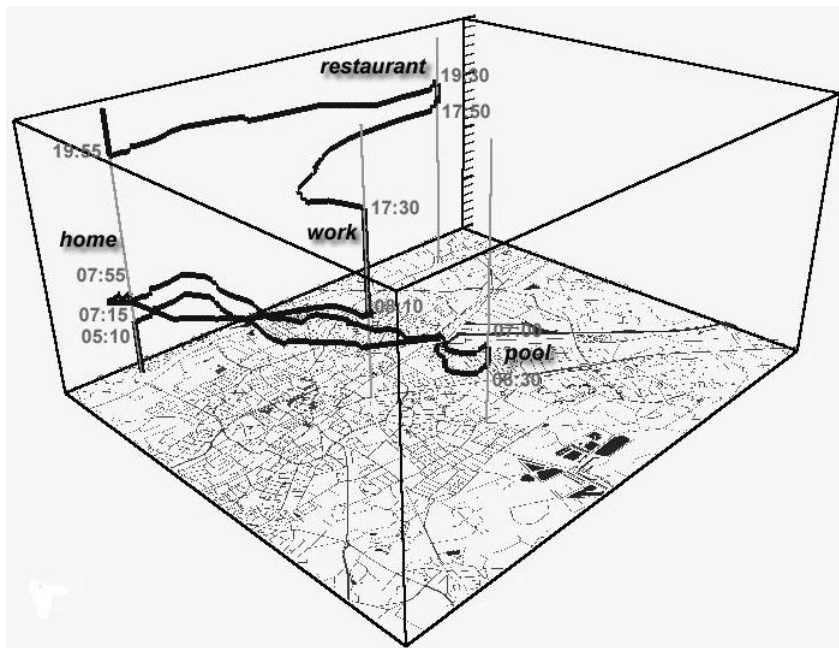


Figure 3. The space-time cube: an example of the person's travels on an average day. The vertical lines indicate a stay at the particular location: home, pool, work and restaurant. The near horizontal lines indicate movements. [17]

The cube contents can be created automatically from a database. Interactiveness enables viewing from any direction [18]. STC facilitates event visualization and analysis. It also supports searching for spatio-temporal patterns.

According to this technique, airport plan is on the bottom of the cube, aircraft horizontal positions are shown by points in three-dimensional space with the vertical dimension corresponding to time. Apart from the trajectories, STC display helps to explore speed: sloping segments indicate fast movement, while steep segments correspond to slow motion.

STC is a proper choice representing a relationship between the horizontal position, time and speed. However, a SKY-Scanner requirement is to visualize the track, distance and altitude. Therefore, STC does not meet the requirements.

3.2. Pure 3D Visualization of Aircraft Trajectories

A strict 3D was used to detect conflicts in the terminal area of Boston Logan Airport [19]; see Figure 4. 3D perspective display shows the situation in a top-down plan-view.

The view can be exocentric, looking at the entire situation from a remote perspective, or egocentric, following an individual aircraft to see the pilot's perspective. Clicking on an aircraft or other object in the environment causes the system to focus the user's view on that object.

Airspace mode represents a set of linked wireframe rings in space (Figure 5). They draw a tunnel in the sky that aircraft appears to fly towards.

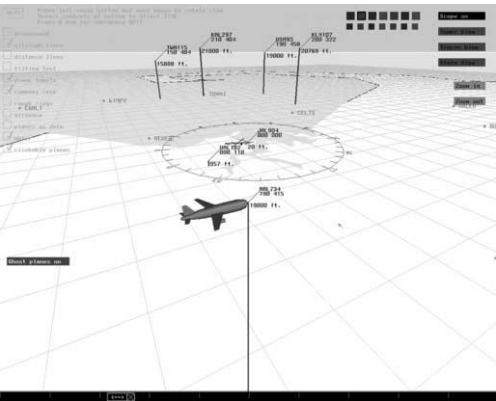


Figure 4. The main view of the controlled zone [18]

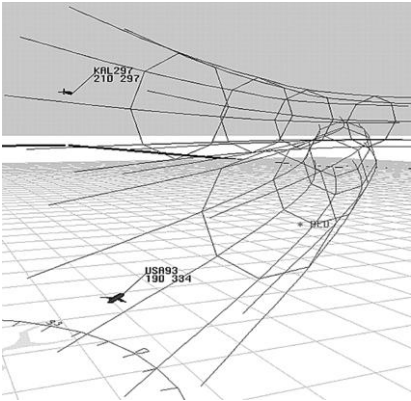


Figure 5. Tunnel in the sky showing the trajectory that aircraft must adhere[18]

This method can be adapted to visualize a relationship between horizontal position, distance and altitude. The tunnel can represent a trajectory prescribed in an assigned instrument approach procedure. The violation is detected when the aircraft is located outside the rings.

3.3. Combined 3D and 2D Visualizations

Combined visualizations enable to show contextual and altitude information at the same time [1]. 3D visualizations serve for integrated attention tasks, such as instructing an aircraft to descend and turn to intercept the localizer. 2D is useful for focused attention tasks, such as estimating the exact aircraft altitude in a moment [16].

2D walls with the projections of an aircraft can be shown in a 3D visualization. The Wall View with Altitude Rulers presents a vertically oriented gradation on the wall (Figure 6). This provides controllers with precise data needed to assess the traffic situation or guide aircraft accurately [7].

The Wall View of Approach Control (Figure 7) enables controllers to check whether the aircraft adheres to the landing/take-off procedure. The view shows unambiguously whether the procedure is strictly followed. This method requires a precise aircraft position that cannot be achieved with the sole radar equipment. Current surveillance technologies cannot implement those visualizations because of inaccuracy of radar devices.

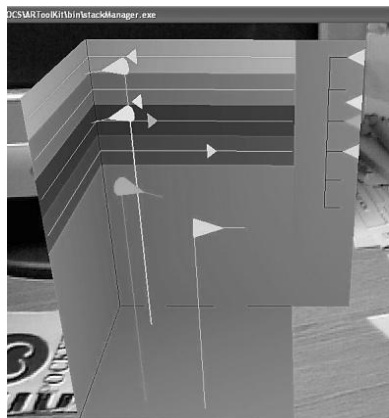


Figure 6. Wall View with Altitude Rulers allows easily read altitudes [7]

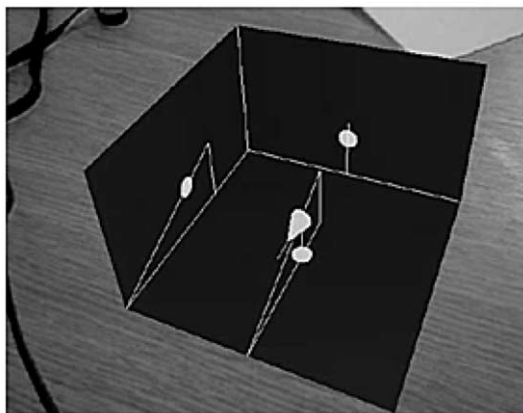


Figure 7. Wall View of Approach Control enables to control adherence to assigned procedure [7]

4. Alternative Prototypes

For evaluation purposes two alternative prototypes are developed. One prototype matches the combination of 2D in 3D, while second – the pure 3D approach.

The SKY-Scanner DSS enables controllers to perceive aircraft position in the context of an assigned airport procedure. Minimizing clutter and distractions is vital to controllers [18]. Hence, on the one hand, it is important to show the actual aircraft position, the assigned procedure and airport terrain. On the other hand, this should be done with minimal means in order not to clutter the screen.

4.1. 2D Curtains in 3D Terrain

Airport procedures present trajectory constraints in a profile view; see an example of approach procedure in Figure 8. This view is convenient for aeronautics professionals. The constraints are presented in alphanumeric texts. The constraints can also be projected in the Wall View with Approach Control (Figure 7). Thus display cluttering is reduced.

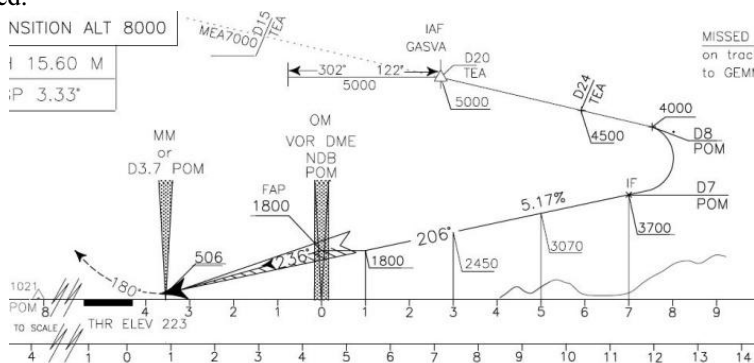


Figure 8. Example profile view of an approach procedure [20]

The rulers enable controllers to monitor a holding stack of landing aircraft. SKY-Scanner integrates two interface models:

1. Wall View with Altitude Rulers
2. Wall View with Approach Control

A benefit of 3D visualizations is in representing 3D ground surface together with altitude. A drawback is that a realistic map may provide little information. Instead, a simple 2D color map based on height can highlight major orientation features [19].

Main window presents a generalized terrain with important for an airport near mountains (Figure 9). Here aircraft are represented with white indicators. Two aircraft are depicted in an example situation: the first is landing and the second is taking off. If the distance is less than allowed minimum, a collision is fixed and the aircraft icon becomes red. In case the minimal safe distance is calculated from predicted positions, the risk evaluation on the DSS control panel becomes yellow and an appropriate message appears on the message board.

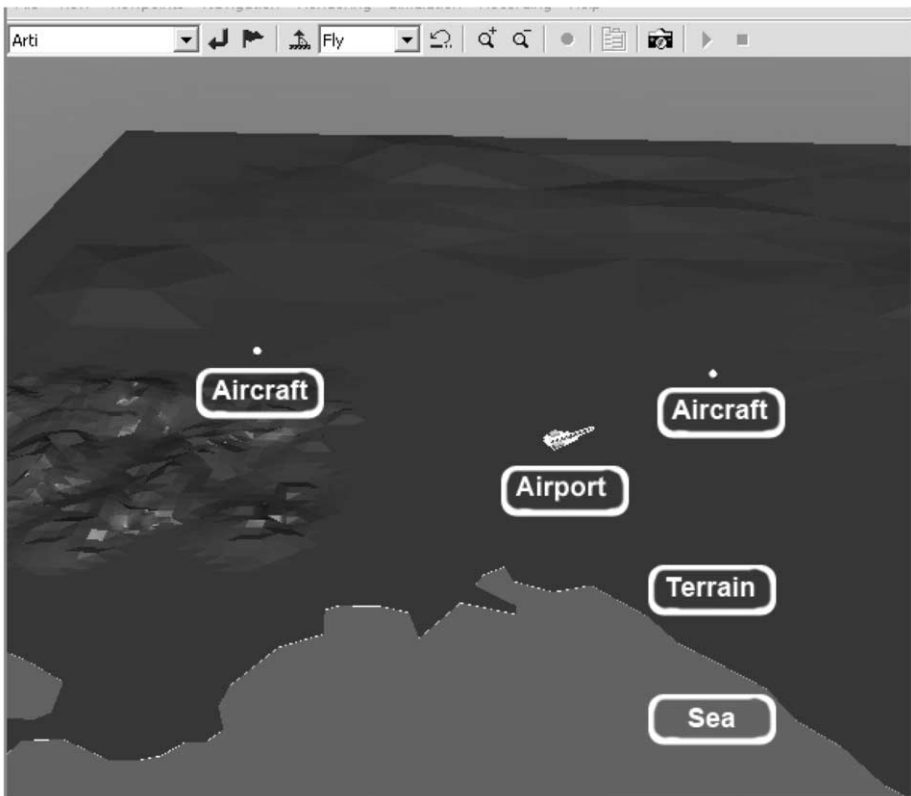


Figure 9. A generalized terrain model. The airport is a white icon in the center. Small white indicators depict two aircraft

Opaque walls from the Wall View with Altitude Rulers (Figure 7) are replaced with transparent curtains (Figure 10). The surroundings are seen like through a curtain. The transition height is represented with a different color. The trajectory is represented with its projection lines on the curtains. Black indicators show projections of an exact aircraft position. Final Approach Fix (FAF) projections are shown with dashed lines.

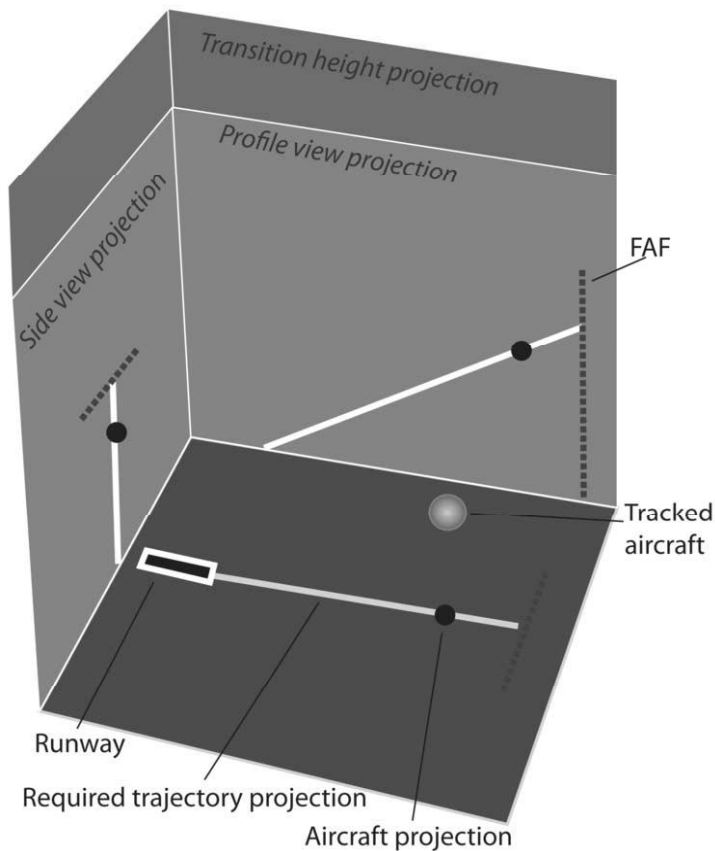


Figure 10. SKY-Scanner visualization model

An airport zone is divided into two vertical spaces. The space below a determined altitude (transitional altitude) is allowed for an aircraft which obtained a landing clearance. Airport procedures are visualized with respect to this space.

In the presented prototype, an airport zone is divided into two areas:

1. A soft control area where the collision risk between the detected aircraft is tracked and the altitude is controlled;
2. A strict control area where an airport procedure is assigned to an aircraft and the constraints (altitude, speed, track) are followed.

In the soft control area the aircraft altitude is observed and longitudinal and vertical distances between aircraft are calculated. The space above a determined altitude is devoted to aircraft which approach the airport from outside. In this space, a controller's task is to ensure appropriate horizontal and vertical separations. Therefore, the altitude rulers are integrated with vertical curtains. The number of rulers depends on waiting loops determined in a concrete airport.

The strict control area is defined within six nautical miles, between Final Approach Fix (FAF) and Touchdown Point (TP). The assigned procedure is presented on the curtains (Figure 11). A green indicator depicts the tracked aircraft. Black indicators on the curtains show aircraft's projections. The transition altitude is presented with a different color on the top of curtains. When aircraft receives a clearance for landing, a DSS support scenario is as follows:

1. Select the assigned procedure. Procedure projections appear on the curtains.
2. Select the tracked aircraft.
3. Observe the situation: aircraft projections have to be on the projection lines.

A path violation can be detected on the projection walls. To adhere to the procedure no deviation from the safe funnel is allowed. The indicators have to follow the projection lines. Path violation risk is visualized with colors:

- green means that risk is evaluated zero,
- yellow means a risk of path violation in the future,
- red means that a violation already occurred. Aircraft position is out of the safe funnel.

The upper part of the control panel shows DSS output. The 3D window attracts the controller's attention when the aircraft deviates from the projection lines. The control panel shows a violated parameter. Risk violation for each tracked parameter is depicted in green, yellow or red and also with a real number from 0 to 1.

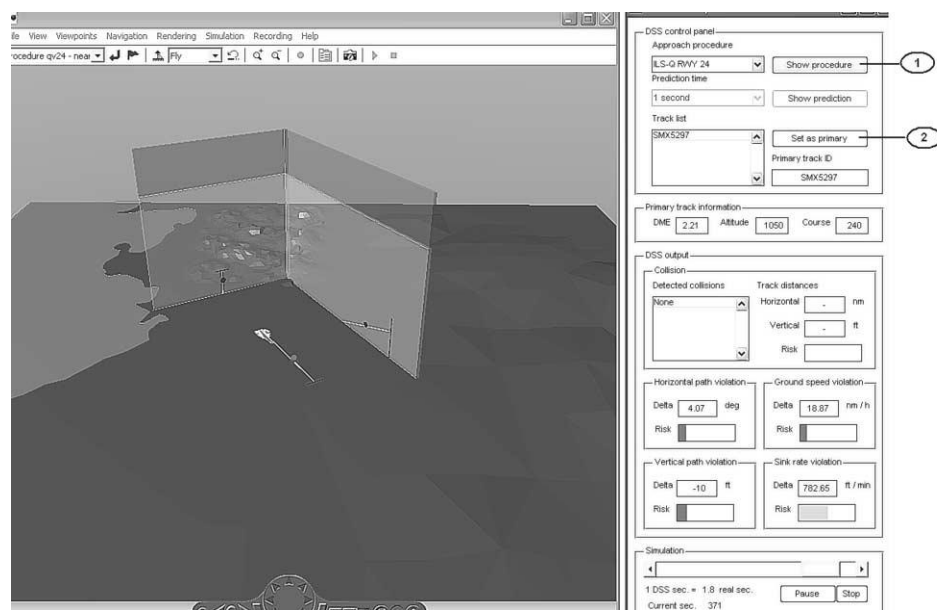


Figure 11. A 3D view with control panel. Button 1 turns on curtains. Button 2 selects the tracked aircraft.

Validity of aircraft position can be easily detected visually and confirmed with colors. The tracked aircraft is depicted in green if it follows the assigned approach procedure. The projections on the walls enable tracking until the touchdown point. The DSS control panel presents current information about the actual tracks including risks. 2D window comprises user interface buttons and a message board.

4.2. Pure 3D Approach

For an alternative prototype a pure 3D approach is chosen. It shows the actual aircraft procedure, airport terrain and the tunnel of the procedure (Figure 12).

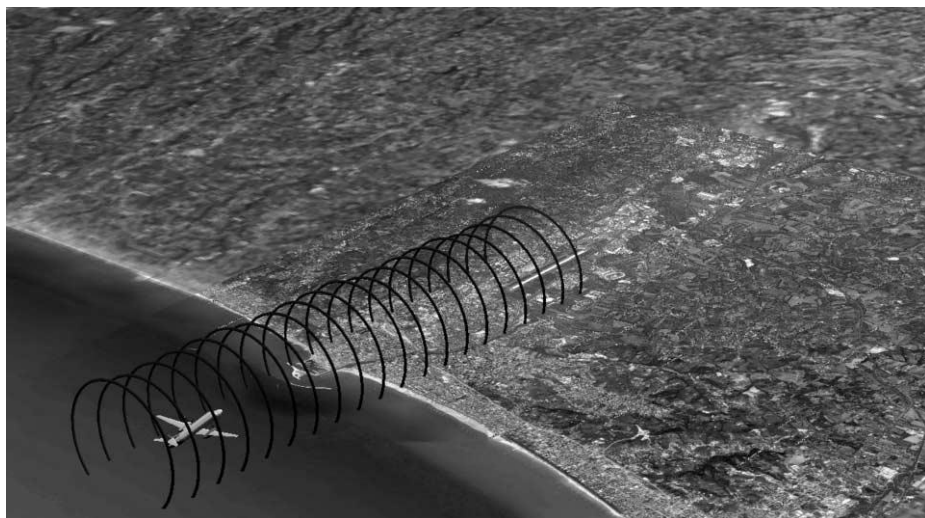


Figure 12. A screenshot of the pure 3D prototype presents a correct aircraft landing

For better visibility an aircraft icon is rather big. An aircraft position within wireframe rings indicates that the airport procedure is adhered to. A photorealistic satellite picture serves as airport terrain. This visualization is less strict than the previous prototype in Figure 11. However it is sufficient to assess the trajectory.

A decision support scenario is analogous to the first prototype. Airport procedure rings appear, when a procedure is assigned to the tracked aircraft. Longitudinal and vertical distances between aircraft are calculated; collision risk is shown with colors. A realistic icon of the aircraft is used in this prototype. This enables to try an alternative visualization and to compare which icon is preferred: realistic or abstract.

5. Conclusions

The paper reports on an ongoing study of a new ATM paradigm. It is based on (1) lidar and radar data fusion, and (2) a 3D visualization of aircraft trajectories within aerodrome traffic zone. The fusion aims at an exact aircraft position. Provided the aircraft position is known, the SKY-scanner DSS can support controllers.

Developed prototypes have been demonstrated to controllers. The feedback highlighted that a combined 2D in 3D visualization is more intuitive and contributes to a better decision support. Generalized terrain has been preferred to photorealistic one. Details of photorealistic terrain have been asserted as distracting and disturbing the observation. Violations have been more distinguishable on the curtains than on the wireframe rings.

A message board with indicators in the first prototype has been evaluated as useful and non-distracting. When the main window indicates a violation, complementary data on the message board present what exactly is violated. In a normal situation there is no need to watch the message board.

A comparison of two prototypes concludes that the prototype with combined 2D in 3D view improves situational awareness. A generalized airport environment depicted with essential terrain obstacles provides sufficient orientation in the environment and

avoids clutter. 3D curtains with projections of an airport procedure reduce cognitive workload of controllers. This enables to estimate the compliance to procedures.

The decision support scenario was judged as realistic and effective while tracking aircraft in the ATZ. The controllers noted that currently they are not able to track aircraft during landing. The developed lidar equipment with DSS provides a means to improve aircraft surveillance also on landing and take-off. This system supports spatial-temporal reasoning in real-time air traffic management tasks.

The prototype is required to develop with MATLAB. This symbolic mathematical tool suits for demonstration purposes. However, another programming tool shall be chosen for industrial implementation. MATLAB is too slow for real time applications. Consider scalability issues.

The decision support scenario can be elaborated to visualize predicted positions and conflicts. The current prototype informs this on the message board.

References

- [1] M. Salerno, G. Costantini, M. Carota, D. Casali. The Sky-Scanner System for Air Traffic. Management: a Simulation Software. *International Journal of Circuits, Systems and Signal Processing*. 4, 1 (2010) 1–8 <http://www.naun.org/journals/circuitssystemssignal/19-209.pdf>
- [2] SESAR Air transport Framework: The performance target. Definition Phase, Milestone Deliverable D2. SESAR Consortium (2006) <http://www.eurocontrol.int/sesar/gallery/content/public/docs/DLM-0607-001-02-00a.pdf>
- [3] SESAR Air Transport Framework: The Current Situation. Definition Phase, Milestone Deliverable D1. SESAR Consortium (2006) <http://www.eurocontrol.int/sesar/gallery/content/public/docs/DLM-0602-001-03-00.pdf>
- [4] Rules of the Air and Air Traffic Services, 13th Edition. International Civil Aviation Organization (1996)
- [5] L. Savičienė. Modeling Path Violation in A Decision Support System FOR Aircraft Approach and Departure. *Proceedings of the 16th International Conference on Information and Software technologies*, IT 2010. Kaunas, Lithuania, 2010, pp. 80-87. ISSN 2029-0020
- [6] J.C. Mankins, Technology Readiness Levels: A White Paper, NASA, Office of Space Access and Technology, Advanced Concepts Office, April 1995. <http://www.hq.nasa.gov/office/codeq/trl/trl.pdf>
- [7] Wong, B.L.W., Rozzi, S., Boccalatte, A., Gaukrodger, S., Amaldi, p., Fields, B., Loomes, M., Martin, P.: 3D-in-2D Displays for ATC. 6th EUROCONTROL Innovative Research Workshop (2007) 42–62 http://inoworkshop.eurocontrol.fr/index.php?option=com_content&view=article&id=32&Itemid=16
- [8] B. Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design* (Interactive Technologies). Morgan Kaufman, 2007.
- [9] G.W.H van Es. Review of Air Traffic Management-related accidents worldwide: 1980 - 2001. Technical report, NLR-TP-2003-376 (2003) <http://www.nlr.nl/smartsite.dws?id=2888>
- [10] R.J. Sternberg, Component processes in analogical reasoning, *Psychological Review*, 84(4) (1977) 353–378
- [11] R. Azuma, M. Daily, Ch. Furmanski. Review of Time Critical Decision Making Models and Human Cognitive Processes. *IEEE Aerospace Conference*, Big Sky, MT, 2006.
- [12] R. Pascual, S. Henderson. Evidence of Naturalistic Decision Making in Command and Control. Caroline E. Zsombok and Gary Klein (Ed.), *Naturalistic Decision Making*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1996.
- [13] D.T. Ogilvie, F.H. Fabian. Strategic Decision Making in the 21st Century Army: A Creative Action-Based Approach. *Fifty-eight Annual Academy of Management*, Stephen J. Havlovic (Ed.), Best Paper Proceedings, CA, August 1998.
- [14] A.M. Mac Eachren, *How Maps Work: Representation, Visualization and Design*. New York: The Guilford Press, 1995.
- [15] R. Azuma, M. Daily, J. Krozel. Advanced Human-Computer Interfaces For Air Traffic Management and Simulation. *American Institute of Aeronautics and Astronautics. AIAA Flight Simulation Technologies Conference* (1996) <http://www.cs.unc.edu/~azuma/AIAA.pdf>

- [16] P. Amaldi, B. Fields, S. Rozzi, P. Woodward, W. Wong. Operational Concept Report, Vol. 1 Approach Control, Vol. 2 Tower Control (No. OCR2-AD4-WP2-MU). Interaction Design Centre, Middlesex University, London, UK, 2005
- [17] M.-J. Kraak. The space-time-cube revisited from a geovisualization perspective. Proceedings of the 21st International Cartographic Conference (ICC) Cartographic Renaissance, Durban, South Africa, (2003) 1988–1996 <http://geoanalytics.net/and/papers/iv04.pdf>
- [18] P. Gatalisky, N. Andrienko, G. Andrienko. Interactive Analysis of Event Data Using Space-Time Cube. Proceedings of the Eighth International Conference on Information Visualization (IV'04), London, England (2004) 145–152 <http://geoanalytics.net/and/papers/iv04.pdf>
- [19] R. Azuma, H. Neely, M. Daily, R. Geiss. Visualization Tools for Free Flight Air-Traffic Management. IEEE Computer Graphics and Applications 20(5) (2000) 32–36 <http://www.cs.unc.edu/~azuma/cga2000.pdf>
- [20] ICAO – Instrument Approach Chart, Napoli/Capodichino, No. 352. ENAV, 2003

Parallel Bidirectional Dijkstra's Shortest Path Algorithm

Gintaras VAIRA and Olga KURASOVA

Vilnius University, Institute of Mathematics and Informatics,

Akademijos St. 4, LT 08663, Vilnius, Lithuania

Gintaras@look.lt, Kurasova@ktl.mii.lt

Abstract. This paper deals with Dijkstra's shortest path algorithm and with the possibilities of speeding-up this algorithm. This algorithm is a breadth-first-search algorithm. The search spreads circularly around the source node in order to find the shortest path from the source node to other nodes. Each following iteration is based on the results of the previous iteration, therefore parallelization of such an algorithm is complicated. The parallel Dijkstra algorithm, proposed in this paper, is based on the bidirectional Dijkstra algorithm and popular multicore technology. Such a technology enables to create a parallel algorithm based on the shared memory and most of the time for data synchronization thereby is saved. The proposed algorithm has been tested using the real road network data. The proposed parallel algorithm is almost 3 times faster than the standard Dijkstra algorithm.

Keywords. Shortest path algorithm, Dijkstra's algorithm, parallel computing, multithreading.

Introduction

At first sight, the shortest path problem seems very simple and global positioning system (GPS) devices and many other systems find the shortest path between two locations quite quickly. When the solution is given within a few seconds, it does not seem very slow and the result is acceptable. However, modern systems deal with much broader route planning tasks. This problem is called a Vehicle Routing Problem (VRP), challenging a combinatorial optimization task, and is widely considered at present. Because of its complexity, this problem is divided into some smaller problems, for example, Vehicle Routing Problem with Time Windows (VRPTW), with Pick-Up and Deliveries (VRPPD), with Satellite Facilities (VRPSF), etc. However, all these problems share the same subproblem - the shortest path problem. In order to solve the logistic task made up of N destination nodes, the $N \times (N-1)$ shortest paths need to be calculated. In the simplest case, the distance between these nodes can be calculated according to the coordinates of nodes. However, the difference between the real shortest path and the straight line can be significant for small distances. For example, if the river exists between two nodes, then the shortest path will increase depending on the nearest bridge, or the shortest path is searched in the city with many one-way roads, and then the search of the shortest path may increase for several times. This task becomes even more complicated, if we take into account the additional detailed road data, for example, a permissible maximum weight of a bridge, and the parameters of the vehicle, or even dynamic information, like traffic jams. If the routes are planned for

K different vehicle types, for example, trucks, cars or even motorcycles, the $K \times N \times (N-1)$ shortest paths need to be calculated. Taking into account all these parameters, the difference between the shortest path between two points and the straight line can be very significant. These examples show that the shortest path search in vehicle routing problems is of great importance. The shortest path problem is still widely considered: new algorithms are being developed and speed-up of the existing algorithms is being considered.

When searching for the shortest path in the road network, a graph with non-negative edges is commonly used. An edge in the graph can be described by any numerical value: distance, time, speed, etc. And a commonly used approach for finding the shortest path in the graph is Dijkstra's algorithm. This paper deals with the speed-up technique of Dijkstra's algorithm. The speed-up method, based on shared memory parallel computing, is proposed and investigated here.

1. Related Works for Speed-Up of Dijkstra's Algorithm

Route planning and shortest path problems have gained more and more attention in recent researches. There are some attempts to develop new algorithms and accelerate the already known ones, by adding a new ingredient or processing additional information. Some of them pay attention to dynamic information, such as road congestion or weather conditions. The others refer to the fact that the path between two points is static, i.e., a graph does not change during the calculation of the shortest path between two nodes, and no additional calculations, based on the traffic condition changes, are made. Such algorithms are simply called static route planning algorithms.

As mentioned before, one of the most popular static algorithms is Dijkstra's algorithm, developed in 1959. It is known as the most efficient algorithm for the shortest path problem in a directed weighted graph. In the paper, we focus on the published results on speed-up techniques of Dijkstra's algorithm. Dijkstra's algorithm is a weighted breadth-first search algorithm. Although this algorithm was designed to calculate the "shortest distance" from one node to other nodes in the graph, it can be easily used for calculating the distance from one node to the destination node. One of the speed-up modification algorithms is a bidirectional Dijkstra algorithm [1], [2], [3], [4]. This method calculates a path starting a search operation from both sides at the same time. The calculation "meets" and stops somewhere in the middle of the road and gives the answer in quest. We will review the mentioned modification more in detail in the next section.

Another attempt to speed-up Dijkstra's algorithm is index usage for the priority queue of labeled nodes. A Fibonacci heap [5] or a binary heap [6] is proposed for indexing a set of nodes. Usage of such an index in a queue of nodes speeds up the algorithm just by one step, extracting nodes with the minimum distance from all the available labeled nodes. Using this technique, the calculation accelerates only in a very large graph, where the count of labeled nodes significantly increases during calculation. However, one-step acceleration does not yield a significant result.

Also, attempts are made to implement Dijkstra's algorithm in reconfigurable hardware. Paper [7] provides an overview of applications of reconfigurable computing in network routing, where a FPGA (field-programmable gate array) - based version of Dijkstra's shortest path algorithm is also presented and differences of the performance between the FPGA-based and microprocessor-based versions of the same algorithm are

compared. Another interesting hardware used for Dijkstra implementation is DAPDNA-2 - dynamically reconfigurable processor developed by IPFlex [8]. The modified algorithm finds the shortest paths in parallel, using the Processing Elements (PE) Matrix. Although the use of the array of 376 PE compared to the microprocessor gives better results, the implemented schema is designed only for DAPDNA-2 and is not suitable for the other hardware.

Since Dijkstra's algorithm is static and calculations are made with a static graph, various preprocessing techniques are used for speeding up the process. One of the easiest methods is to count the shortest paths between all N nodes [9]. The obtained $N \times N$ matrix then can be easily used in the next level route planning system. However, the use of such an approach together with road data of the real world would be very inefficient. Great speed-up factors can be achieved using highway hierarchies [4], [10]. The highway hierarchy method is based on the idea that only a highway network needs to be searched outside a fixed size neighborhood around the source and target. A highway approach is faster in preprocessing as compared to the Arc-lags approach, but calculates the shortest path more slowly [4]. The main "arc-flag" method idea is a graph partition into rectangular regions. Then, all the edges are reviewed by marking with property flags, which indicate whether the edge is on the shortest route to the regions or not. During the route search, only those edges are selected the properties of which are appropriate, and the rest are rejected.

Paper [2] also examines other graph partitioning techniques: quadtree, kd-tree, and two-level partitioning. According to the test results, the author suggests using the kd-tree method, because it leads to the best results and is easy to implement [2]. The splitting technique can be used to form a second-level graph, i.e., the graph is split into parts, which together make a new graph of macronodes and the macronodes are comprised of smaller graphs [9]. Thus, Dijkstra's algorithm would be first used in the macronode graph, and then, according to the obtained results, it would be used in other smaller graphs. All of these preprocessing technologies give good results, but they also have disadvantages: each preprocessing technique requires additional data storage for the edge or node, such as in the arc-flag method, where each node keeps property flags about all regions. Thus, for a very huge graph (for example, OpenStreetMap data is made up of ~600M nodes), we should have to deal with memory problems, or will reduce the algorithm efficiency using a hard disk and reading it constantly. Another drawback is a difficult implementation using dynamic data, such as roads closed for repair, or congestion, etc. Then all the partitions will have to be preprocessed once more, which may last very long for a very large graph. Another important aspect is that the preprocessing of the data is performed through the calculation of distances: division is made by taking into account geographical features of the road. Since Dijkstra's algorithm may calculate the route using the edge cost, which may include not only the distance, but also the time or other values, preprocessing methods lose their value.

Another useful technology to speed up Dijkstra's algorithm is parallel computing. The approach has already been mentioned in adapting Dijkstra's algorithm to reconfigurable hardware. To speed up the preprocessing part, the $N \times N$ road calculation is proposed using the parallel computing. To calculate all roads, the usage of total \sqrt{N} processors is proposed [9], [11]. However, this method is still very limited, even with a modern technology. It is suitable to use more in local computer networks, which usually covers a smaller physical area, like home, office, or a small group of buildings, than in street routes. Dijkstra's algorithm is iterative and, in each iteration, it uses the data obtained in the previous iteration. Due to these properties, the algorithm cannot be

easily adapted to parallel computing, but it is still widely considered. Usually, due to the large amount of data, it is difficult to adapt Dijkstra's algorithm to the distributed memory parallel computing. However, today's multicore technology allows us to easily implement parallel calculations based on a shared memory technique, called Transactional Memory (TM), thus avoiding synchronization of large amounts of data. TM is a new technology in multicore platforms that allows several different processes to access the same memory location. The developer has a possibility to mark certain parts of the code, indicating that during the program execution at this point, some memory allocations can be accessed by several different processes. TM monitors process the transactions, and if several processes are trying to access the conflicting memory, TM decides how to handle [12], [13]. In general, all the processes are blocked, only one process can access the memory allocation and, when the operation is completed, the blocked process is again continued. If the transactions are not conflicting, processes are carried out without any interruption. Because of their properties TM is very useful for parallel computations, based on the shared memory. There are some attempts to implement Dijkstra's algorithm in parallel computing using the Transactional memory together with Helper Threads [11], [12], [13]. Such a parallel computing technology is proposed to use in the inner loop of Dijkstra's algorithm, where nodes are processed for labeling. The helper thread reads the tentative distance of the vertex in the queue and attempts to relax its outgoing edges based on this value [13]. When the processes are finished, the main process continues its work up to the next inner loop. It is also proposed not to wait to the end of the helper processes, and the main process continues to work without paying attention to the adjacent processes [12], [13]. Such an approach cannot work properly without TM technology. However, such helper thread computing can last shorter than its starting and termination. The periodic creation and destruction of such processes could adversely affect the operation of the algorithm.

Paper [14] gives a similar parallel computing method, but introduces an additional heuristic, for example, choosing which edges need to be dismissed. Also, attempts are made to adapt parallel computing to algorithms that are based on preprocessing. In a modified arc-flag algorithm, a linear preprocessing method was left, but only the shortest path searches are performed in parallel [3]. The parallel computing is possible without TM technology, and then each new process will have to keep in memory a separate copy of the road data [11]. However, bearing in mind the size of the modern road networks, problems can arise due to technological limitations while having multiple copies of the same graph in memory. Although this method can be easily implemented in parallel with a separate memory, this method requires an additional synchronization in order to update data everywhere.

Dijkstra's algorithm acceleration and parallelization still remains a complex problem and completely unsolved. In order to speed-up this algorithm, several methods together are often used, for example, Fibonacci heap and highway hierarchies. However, in order to speed up Dijkstra's algorithm, the main idea of this algorithm is often distorted: it is the "shortest" path search algorithm. And many of speed-up methods lead to nearly the shortest path calculation, such as heuristic introduction or highway hierarchies.

2. Dijkstra's Algorithm

Dijkstra's algorithm finds the shortest distance from one node to all the others in the graph with non-negative edges. Let us consider a graph $G = (V, E)$, which consists of the nodes of $v \in V$ and edges $e \in E$. Let us denote by $d(v)$ the distance from the node $v \in V$ to the starting node, and by $w(v, u)$ the distance value of the edge from node u to node v . All the labeled nodes are organized by the algorithm in the priority queue Q and all the visited nodes are stored in array S . During each iteration the algorithm extracts node k from queue Q with the lowest value of $d(k)$. Then all the outgoing edges of node k are relaxed, which could reduce the keys of the corresponding neighbors. Relaxing an edge (k, v) means testing whether we can improve the shortest path to v found so far by going through node k . So, if $d(k) + w(k, v)$ is less than $d(v)$ found so far, $d(v)$ is replaced by a new value. If the adjacent nodes have not yet been labeled, they are inserted in queue Q . This operation is performed in the decreaseKey operation [1], [2], [8], [12]. The following pseudo-code illustrates Dijkstra's algorithm. The algorithm terminates when the destination node is found.

```

procedure Dijkstra( $G = (V, E), s, d$ )
begin
   $d(s) := 0$ 
   $S := \{ 0 \}$  /* Visited nodes */
   $Q := \{ 0 \}$  /* Labeled nodes */
   $Q.insert(s, 0)$ 
  while (! Empty( $Q$ )) do begin /* OUTER LOOP */
     $u := Q.extractMin()$ 
     $S.addNode(u)$ 
    if  $u == d$  then /* STOP CRITERION */
      break; /* route found */
    for each  $v$  adjacent to  $u$  do begin /* INNER LOOP */
       $sum := d(u) + w(u, v);$ 
      if ( $d(v) > sum$ ) then begin
         $Q.decreaseKey(v, sum);$ 
         $d(v) := sum;$ 
         $pi(v) := u;$  /* set predecessor */
      end
    end
  end
end
end

```

Dijkstra's algorithm is a labeling algorithm. When the distance from the current node to the start node is known, adjacent nodes then are labeled. So by starting labeling of the nodes adjacent to the start node, the algorithm iterates until the set of labeled nodes is empty. Dijkstra's algorithm is a greedy algorithm because at each step, the best alternative is chosen. The algorithm produces a correct shortest-paths tree whose top is the start node, and to every other node in the graph G there is only one possible path. So, while executing the search from the start node s to the target node t , the algorithm must visit all the nodes $v, v \in V$, with the distance $d(v) < d(t)$.

2.1. Bidirectional Algorithm

In search of the path between two specific nodes of the graph, a modified Dijkstra algorithm - bidirectional method - can be used. This method performs the searches starting from the start and end nodes [1], [2], [3]. The algorithm is simply run by executing one step on each side in a single period. At first, the processing step with the extracted node is performed from the start node, and then the same calculation is made from the end node. During such a step a single node is extracted from the priority queue, marked as visible and all the corresponding edges are relaxed (inner loop). Such a process will not necessarily be symmetrical. It will depend on the number of edges of all the visited nodes.

To execute such an algorithm, separate data containers must be used, so each search must have its own sets for labeled and visited nodes. In the following pseudocode, a forward search is using the priority queue Qs and the set of visited nodes Ss and a backward search is using the priority queue Qd and the set of visited nodes Sb.

```

procedure bidirectionalDijkstra(G = (V,E),s,d)
begin
  (Qs,Ss):=createEmptyContainers();/*from start*/
  (Qd,Sd):=createEmptyContainers();/*from end*/
  ...
  while (!Empty(Qs) and !Empty(Qd) ) /* OUTER LOOP */
  do begin
    /* calc from start */
    u := Qs.extractMin()
    Ss.addNode( u )
    ...
    if (stopCriterion() is true) /* STOP CRITERION */
      break;
    for each v adjacent to u do begin /* INNER LOOP */
      ...
    end
    /* calc from end */
    u := Qd.extractMin()
    Sd.addNode( u )
    ...
    if (stopCriterion() is true) /* STOP CRITERION */
      break;
    for each v adjacent to u do begin /* INNER LOOP */
      ...
    end
  end
end
end

```

The bidirectional algorithm stops when the stopping criterion (stopCriterion()) is met. This occurs somewhere in the middle between the start and end nodes. Since the bidirectional Dijkstra algorithm is two-sided and the search is carried out from the start node and from the end node at the same time, it is necessary to establish clear stop criteria. An algorithm without such criteria will be run as long as the search from the

start node will find the end node or the search from end node will find the start node. In this case, there will be a lot of nodes that will be visited twice by contrariwise computations. So there will be such a set $H = S_s \cap S_b$, and the set H will consist of all the twice visited nodes. To increase the efficiency of the bidirectional Dijkstra algorithm, the set H needs to be minimized. Papers [1], [2], [3] present two possible stop criteria: when the current labeled node has already been labeled by the other search and when the current visited node has already been visited by the other search. With the first stop criterion the algorithm stops when such a node w is found, where $w \in Q_s$ and $w \in Q_b$. This stopping criterion can be defined as the intersection of two sets: $Q_s \cap Q_b = \{w\}$. If we denote the shortest path from the node v to the start node by $P(v)$, $v \in V$, and the shortest path from the node v to the end node by $P(u)$, $u \in V$, we get the shortest path $P(w) \cup P(w)$ with the first stop criterion. Application of this stop criterion to the search in the example graph is illustrated in Figure 1. The shortest path is ACB.

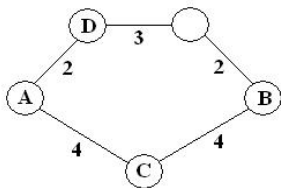


Figure 1. The bidirectional algorithm chooses the shortest path ACB by applying the first stop criterion.

The algorithm stops by applying the second stop criterion, when the node z is found, where $z \in S_s$ and $z \in S_b$. This stopping criterion can be defined as the intersection of two sets: $S_s \cap S_b = \{w\}$. Then the shortest path from the node s to the node t is $P(z) \cup P(z)$. Figure 2 illustrates the example graph, in which we get the wrong result by applying the second stop criterion. It happens because $S_s \cap S_b = \{E\}$, and the nodes B and C remain just labeled by different searches.

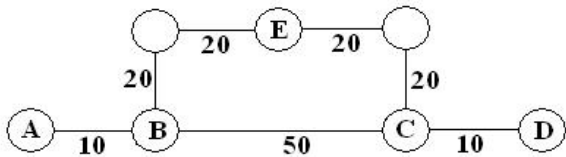


Figure 2. The bidirectional algorithm chooses the path ABECD by applying the second stop criterion.

By applying the second stop criterion to the first example, we also get the “wrong shortest” path, however, in the second example, the difference between the correct and wrong shortest paths is more obvious. By applying the second stop criterion to the second example, we get the shortest path ABECD instead of ABCD. It happens because there exists such a node E , where $w(B,E) < w(B,C)$ and $w(C,E) < w(B,C)$. In the worst case, the path BEC can be almost twice longer than the path BC. Such a situation can lead to a really significant inaccuracy.

3. Proposed Scheme

3.1. Modified Bidirectional Algorithm

As we have already mentioned, Dijkstra's algorithm was developed to calculate the shortest path. As we have seen, the bidirectional algorithm, and many other Dijkstra's algorithm modifications, cannot find the shortest path. Sometimes the error of calculation can be significant. Before modifying Dijkstra's algorithm to speed it up, we have set a requirement that the new algorithm cannot provide a different answer than the standard Dijkstra algorithm. The modified algorithm must find the exact shortest path. The proposed parallel Dijkstra algorithm is based on the bidirectional method. Since the standard bidirectional algorithm does not meet the requirement that we had set, the first step is to modify the bidirectional algorithm. To achieve accurate results by means of the bidirectional algorithm, we combine both stop criteria. The algorithm is extended with a set R of possible answers and with a finish phase. If the visited node of contrariwise computation is found for the first time (found node z , where $z \in S_s$ and $z \in S_b$), then the algorithm enters the finish phase. In this phase, the process continues without appending the priority queue with new labeled nodes, i.e., without the inner loop. After completing these additional steps, we get alternative nodes z_i , $i=1..N$, where $(S_s \cup Q_s) \cap (S_b \cup Q_b) = \{z_i, i=1..N\}$. During such a process other alternative routes are found and added to the set R . Thus, after finishing such a process, the set $R = \{ P(z_1) \cup P^l(z_1), P(z_2) \cup P^l(z_2) .. P(z_N) \cup P^l(z_N) \}$. The calculation ends, when the priority queue Q remains empty. Then the shortest path can be extracted from the set R of possible answers. The following pseudocode implements the proposed method.

```

procedure biDijkstra( $G = (V, E), s, d$ )
begin
  Boolean finish := false;
   $R := \{ \emptyset \}$  /*possible answers list*/
  ...
  while (!Empty( $Q_s$ ) and !Empty( $Q_d$ )) /* OUTER LOOP */
  do begin
    /*PERFORM FORWARD SEARCH*/
     $u := Q_s.extractMin()$ 
     $S_s.addNode(u)$ 
    ...
    if ( $u \in S_s$  and  $u \in S_b$ ) /* STOP CRITERION */
    then begin
      finish:=true; /* set finish phase */
       $R.addAnswer()$ ;
    end
    if (not in finish phase ) then begin
      for each  $v$  adjacent to  $u$  do begin /*INNER LOOP*/
        ...
      end
    end
    /*PERFORM BACKWARD SEARCH*/
  end
  ...

```

```

end
R.findShortest(); /* get shortest path from list */
End

```

We get $(S_s \cup Q_s) \cap (S_b \cup Q_b) = \{E, B, C\}$ according to this termination condition, where the example, presented in Figure 2, is analyzed. And we get the set $R = \{P(E \cup P^l(E)), P(B) \cup P^l(B), P(C) \cup P^l(C)\}$. Next we can compare the obtained results $|P(E) \cup P^l(E)| = 100$, $|P(B) \cup P^l(B)| = 70$, $|P(C) \cup P^l(C)| = 70$. By selecting the smallest of these values we get the shortest path ABCD. In the first example, when solving the shortest path problem in the same way, we get the shortest path ADB. The modified bidirectional Dijkstra algorithm finds the same path as the standard Dijkstra algorithm. Therefore, the requirements have been met. The bidirectional Dijkstra algorithm is 2 times faster than the standard Dijkstra algorithm [1], [2], [3]. The modified bidirectional algorithm lasts slightly longer because of the finish phase. However, the algorithm is still faster than the standard Dijkstra algorithm.

3.2. Parallel Algorithm

The scheme of the parallel algorithm, proposed by us, is essentially based on the modern multicore technology. The modified bidirectional Dijkstra algorithm is transformed into a parallel form, based on a shared memory technology. For this method, an ordinary two-core processor is required. Thus, the two processes may look for the shortest path from the start and end nodes at the same time. Like in the modified bidirectional Dijkstra algorithm, each process must access data of the contrariwise computation to check for stop criteria. If the stop criterion is reached, then the process proceeds to the finish phase and “tells” the opposite process to do the same. Both processes fill out the set R of possible answers with the shortest paths found. The processes are stopped, when the priority queue Q becomes empty. Then the main process selects the shortest path from set R.

```

procedure parallelDijkstra(G = (V,E),s,d)
begin
  Boolean finish := false;
  R := { 0 } /*possible answers list*/
  ...
  /* call backward process */
  calcBackwardsDijkstra(G,s,d,R);
  ...
  while (!Empty(Q)) /* OUTER LOOP */
  do begin
    ...
    u := Qs.extractMin();
    Ss.addNode( u );
    if (u ∈ Ss and u ∈ Sb) /* STOP CRITERION */
    then begin
      finish:=true; /* set finish phase */
      R.addAnswer();
    end
  end

```

```

if( not in finish phase ) then begin
  for each v adjacent to u do begin /*INNER LOOP*/
    ...
  end
end
end
waitForOtherProcess();
R.findShortest();
end

```

Since this algorithm is based on parallel shared memory computing, it allows avoiding the additional data transfer, which is necessary in the separated memory parallel technology. However, the problem of sharing data in memory needs to be solved, because the access to shared data cannot be achieved in an uncontrolled fashion. In this paper, we have already mentioned about the Transactional Memory technology by which this problem can be solved. However, this problem can be solved also by means of other technologies, for example, Mutual Exclusion (mutex) directives. Such directives in the program can ensure that only one process is executing an operation protected by the mutex object. This paper does not explicitly deal with the efficiency of the mentioned technologies or other similar technologies. However, no matter which technology is chosen, we need to identify certain parts of the algorithm, which cannot be carried out simultaneously by two separate processes. By identifying those areas of algorithm we can guarantee that the two processes at the same time will not operate the data at the same memory location. In the proposed algorithm, such areas are the sets S and R. The parts of the algorithm in which these two sets used are either read or modified are as follows:

```

S.addNode(u)
...
if (u ∈ Ss and u ∈ Sb) /* STOP CRITERION */
...
R.addAnswer();

```

Those identified parts of the algorithm need to be synchronized so that only one process at a time could perform the operation, and one process at a time would be blocked until another process ends the transaction. However, the other part of the code will run in parallel.

The main disadvantage of this algorithm is that it cannot be adapted to a larger number of processes. However, an additional process will be started and destroyed only once and will take the most time during the calculation. This allows us to avoid a number of additional delays that occur in the startup and destruction of an additional process.

4. Experimental Evaluation

This parallel algorithm was implemented with pthread - a POSIX standard for threads. Each process of the proposed parallel approach is implemented as a separate thread. In order to solve conflicted access to the same memory area, we used "mutexes" - pthread

algorithms that are used in concurrent programming to avoid the simultaneous use of the common resource by pieces of the computer code called critical sections. The algorithm was developed in the C language and tested in the latest 64-bit Fedora Linux Operating System. The experiments were carried out using Dell Vostro 1400 laptop with 667 Mhz 2Gb memory and the Intel Core 2 Duo T7250 2GHz processor with Bus speed of 800MHz. The algorithm was tested on a real road network using the OpenStreetMap data. Figure 3 provides examples that were visualized with the Mapserver open source software.

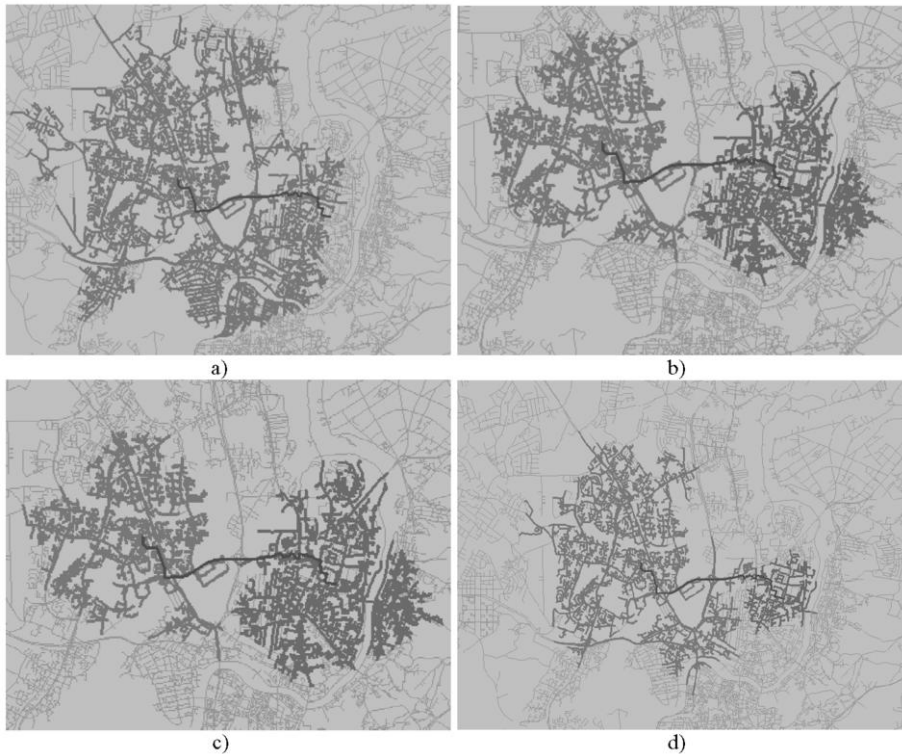


Figure 3. The shortest path calculation: a) using the standard Dijkstra algorithm; b) using the modified bidirectional Dijkstra algorithm; c), d) using the proposed parallel scheme.

The steady reading of data from the disk can cause delays, which distorts the results. In order to avoid that, a Lithuanian road network has been selected and placed in the computer memory before testing. Five random nodes A, B, C, D and E were selected in the graph in the same city and all the shortest paths between them were calculated. The tests were calculated by the standard Dijkstra algorithm (Figure 3a), the modified bidirectional Dijkstra algorithm (Figure 3b), and by the parallel algorithm proposed (Figure 3c). Figure 3d shows the shortest path calculation, where the second process delayed to start because of the operating system loads. The test results are presented in Figures 4 and 5. These results indicate that the modified bidirectional algorithm is still 2 times faster than the standard Dijkstra one. However, the parallel algorithm is almost 2.9 times faster than the standard Dijkstra one and 1.4 times faster

than the bidirectional Dijkstra algorithm. So the efficiency $E(p)$ of the proposed parallel algorithm is 0.7 ($p=2$).

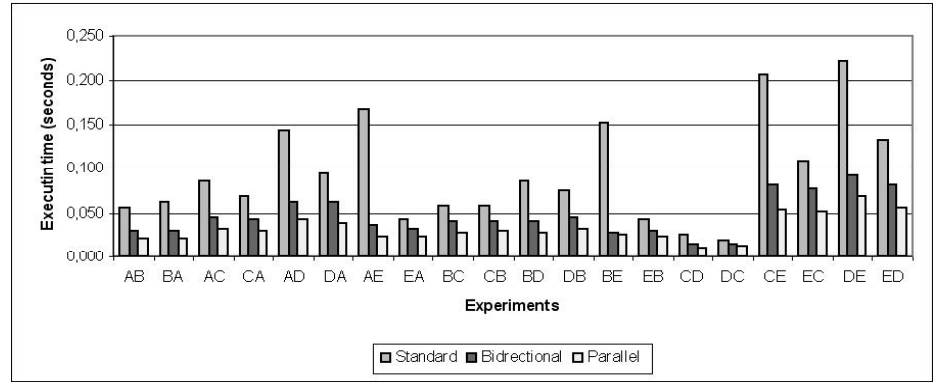


Figure 4 . Execution time of the calculation of the shortest path between nodes A B C D E.

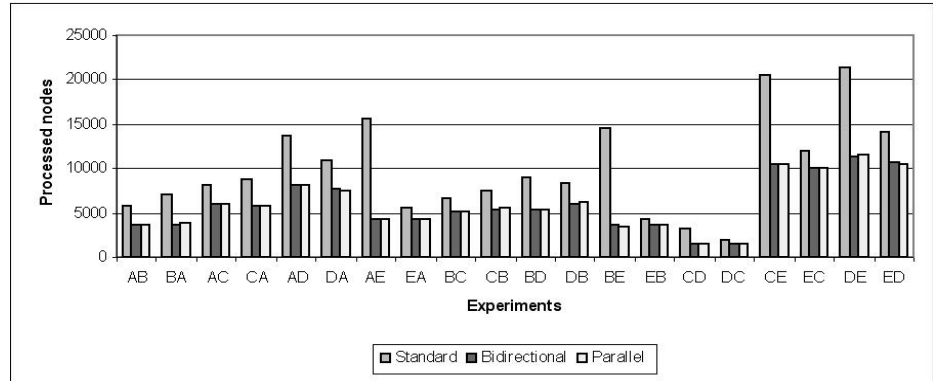


Figure 5. The number of processed nodes in the calculation of the shortest path between nodes A B C D E.

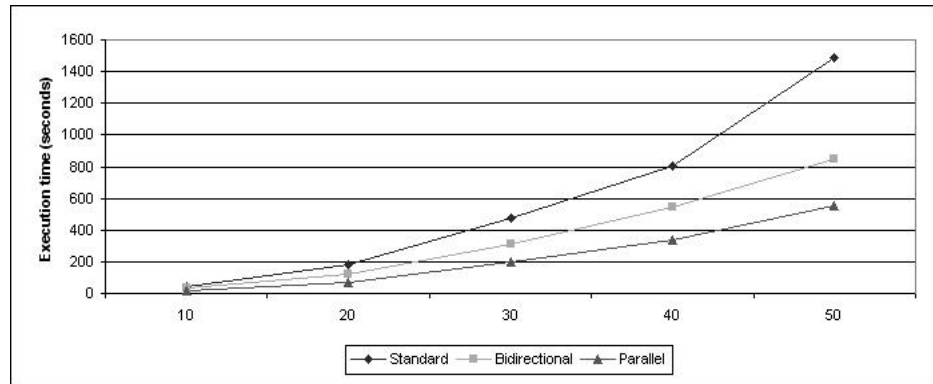


Figure 6. Execution time of the calculation of all the shortest paths between random N nodes.

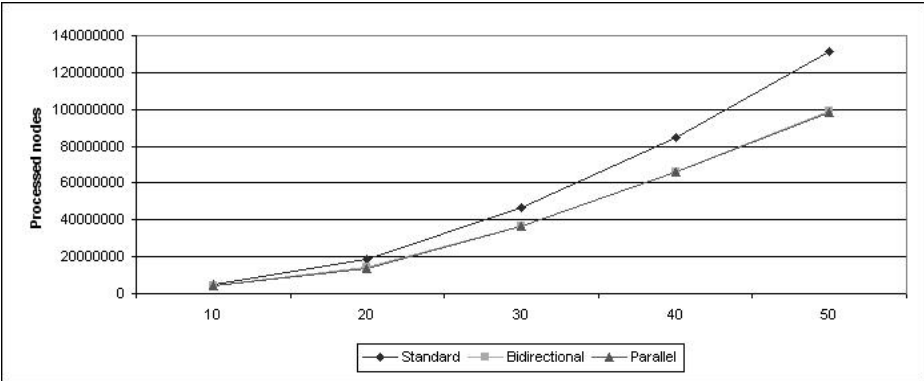


Figure 7. The number of processed nodes in the calculation of all the shortest paths between random N nodes.

Table 1. Execution time and the number of processed nodes in the calculation of all the shortest paths between random N nodes

N	Execution time (seconds)			Processed nodes		
	Standard Dijkstra	Bidirectional	Parallel	Standard Dijkstra	Bidirectional	Parallel
10	47.56	33.68	21.02	5271562	4163721	4244089
20	184.19	122.50	73.34	18511173	14073612	13961447
30	477.07	309.28	200.67	46566414	36361419	36652408
40	800.10	546.23	333.48	84806593	65769961	65779506
50	1488.55	847.68	553.10	131125250	98978796	98404846

Table 2. Average execution time and average numbers of processed nodes

	Execution time	Processed nodes
Standard Dijkstra	0.536	53738.97
Bidirectional	0.350	41530.77
Parallel	0.219	41671.60

In the second experiment, the total time of the shortest path calculations between all N randomly selected nodes was measured. The experiments were done by using the whole road network of Lithuania. Test results are presented in Table 1, Figure 6 and Figure 7. The results show that the calculation of all the shortest paths between all randomly selected 50 nodes lasted ~24 minutes on the same hardware. Meanwhile, the parallel Dijkstra algorithm calculates the same shortest paths ~9 minutes. The average execution time and the average numbers of processed nodes are presented in Table 2.

5. Conclusions

A parallel algorithm based on Dijkstra’s algorithm for the search of the shortest paths has been proposed in this paper. A requirement has been set for the modified speed-up algorithm: the algorithm should find the exact shortest path in the graph. The proposed scheme completely complies with the requirement and finds the shortest path in the graph almost 3 times faster than the standard Dijkstra algorithm. The proposed scheme is based on modern multicore processors and the shared memory parallel computing technology. This algorithm has been successfully implemented in a personal computer

with the Linux operating system, using the pthreads technology. The problem of the conflicted access to the memory area has been solved, using the pthread mutex technology.

Computing times of a single shortest path search obtained experimentally are quite short, less than one second, so the achieved speed-up may not look very effective in the real world. However, the results obtained in the second experiment show that the speedup (~3 times) may look really significant in the implementation of this algorithm, when more complicated problems are solved, such a searching for the shortest paths between all the N nodes for logistic purposes.

Since the algorithm provides the exact answer, as does the standard algorithm, this method can change Dijkstra's algorithm in other areas, such as computer network algorithms. The scheme proposed may also be combined with the other Dijkstra speed-up techniques, reviewed in this paper. However, the parallel algorithm has disadvantages as well. It is limited to two parallel processes. Thus, a further research should be directed to the Dijkstra algorithm acceleration in computers with more than two processor cores.

References

- [1] Goldberg, A.V., Kaplan, H., Werneck, R.F.: Reach for A*: Efficient Point-to-Point Shortest Path Algorithms. In Workshop on Algorithm Engineering and Experiments (ALNEX) (2006), 129–143.
- [2] Schutz, B.: Partition-Based Speed-Up of Dijkstra's Algorithm, 2008.
- [3] Berrettini, E., D'Angelo, G., Dellinger, D.: Arc-Flags in Dynamic Graphs, 2009.
- [4] Koehler, E., Moehring, R. H., Schilling, H.: Fast Point-to-Point Shortest Path Computations with Arc-Flags. In Proc. of 9th DIMACS Implementation Challenge - Shortest Paths, 2007.
- [5] Von Lossow, M.: A min-max version of Dijkstra's algorithm with application to perturbed optimal control problems. In Proc. in Applied Mathematics and Mechanics (PAMM), 2008.
- [6] Madduri, K., Bader, D. A., Berry, J. W., Crobak, J. R.: An Experimental Study of a Parallel Shortest Path Algorithm for Solving Large-Scale Graph Instances. In Proc. 9th Workshop on Algorithm Engineering and Experiments (ALENEX 2007), 2007.
- [7] Tommiska, M., Skytta, J.: Dijkstra's Shortest Path Routing Algorithm in Reconfigurable Hardware. In Proc. of the 11th Conference on Field-Programmable Logic and Applications (FPL 2001), Belfast, UK (2001), 653–657.
- [8] Ishikawa, H., Shimizu, S., Arakawa, Y., Yamanaka, N., Shiba K.: New Parallel Shortest Path Searching Algorithm based on Dynamically Reconfigurable Processor DAPDNA-2. In Proc. of ICC'2007, (2007), 1997–2002.
- [9] Romeijn, H. E., Smith, R. L., Parallel Algorithms for Solving Aggregated Shortest Path Problems. Computers & Operations Research 26 (1999), 941–953.
- [10] Knopp, S., Sanders, P., Schultes, D., Schulz, F., Wagner, D.: Computing Many-to-Many Shortest Paths Using Highway Hierarchies. In Workshop on Algorithm Engineering and Experiments (ALENEX) (2007), 36–45.
- [11] Chang-le Lu, Yong Chen: Using Multi-Thread Technology Realize Most Short-Path Parallel Algorithm, 2006.
- [12] Anastopoulos, N., Nikas, K., Goumas, G., Koziris, N.: Employing Transactional Memory and Helper Threads to Speedup Dijkstra's Algorithm. In Proc. of International Conference on Parallel Processing (ICPP), 2009.
- [13] Anastopoulos, N., Nikas, K., Goumas, G., Koziris, N.: Early Experiences on Accelerating Dijkstra's Algorithm Using Transactional Memory. In Workshop on Multithreaded Architectures and Applications (MTAAP), 2009.
- [14] Edmonds, N., Breuer, A., Gregor, D., Lumsdaine, A.: Single-Source Shortest Paths with the Parallel Boost Graph Library. In The Ninth DIMACS Implementation Challenge, 2006.

This page intentionally left blank

Subject Index

2D and 3D visualization	408	household	353
Ådalsbanan	271	human-computer interaction	380
air traffic control	408	hypergraph	108
annotations	66	inductive learning	380
archetypes and archetype patterns		information security	353
based development	283	information systems	257
aspect	197	information technology	325
awareness	353	integrated	325
BPEL	183, 240	interactive inductive learning	380
BPMS	240	ISO/IEC 15504	296
business process modeling (BPM)	169	IT GRC	325
capability maturity model		living enterprise	257
integration (CMMI) for		machine learning	380
development	283	mappings	139
CEP	240	model driven architecture (MDA)	55, 169, 283
class-based modeling	125	model transformations	37, 55, 66, 94
CMMI	296	model-driven security	339
code generation	37	MOF metalevels	125
compilers	94	MOLA	66
compliance	325	multithreading	422
controllers' needs	408	n-ary associations	108
course comparison	380	ontology verbalization	3
data repository	125	OWL ontologies	139
Dempster-Shafer theory	394	parallel computing	422
departure and approach		parallel input layers	369
procedures	408	pattern matching	66
development of LBS	3	personalized LBS	3
digital preservation	271	plan-driven methods	296
Dijkstra's algorithm	422	PMBok	296
domain engineering	283	PMIS	213
domain specific languages		PMIS configuration	213
(DSL)	169, 197	process model	325
domain-specific	66	project management	213, 296
EDRMS	271	project management information	
electronic recordkeeping	271	systems	213
enterprise modeling	257	QoS	227
extreme programming (XP)	283	QVT	55
fair division	394	random spikes	369
geo-spatial ontology	3	RCC8	153
governance	325	RDF	139
grammar	197	RDF data	125
graphical tool building	94		

records continuum model	271	spiking activity propagation	369
redefinition	108	statistical machine translation	153
REDO	80	subsetting	108
redundancy	94	testing	309
relational databases	139	transformation	183
requirements	37	transformation-driven architecture	
responsibility	353	(TDA)	80
reverse geo-coding	3	UMLsec	339
risk management	325	UNDO	80
role-based access control	339	UNDO metamodel	80
scrum	296	UNDO/REDO mechanism	80
SecureUML	339	use cases	37
security modelling languages	339	value-based software engineering	257
self-organization	257	vehicle routing	227
self-testing	309	views on metamodels	94
shortest path algorithm	422	web services	227
simulation	183	WS-notification	240
situational awareness	408	WSRF	240
smart technologies	309	XCPM	213
software development process	55	XML schema	213
spatial ontology	153	YAWL	183
specialization	108	Zachman framework (ZF)	283
SPEM	55		

Author Index

Anderson, K.	271	Luts, M.	3
Barzdins, J.	v	Matulevičius, R.	339
Bērziša, S.	213	Medvedis, I.	169
Bicevskis, J.	169, 309	Murane, I.	353
Bildhauer, D.	108	Nael, M.	296
Birzniece, I.	380	Nesterovs, S.	169
Blumbers, N.	240	Nikiforova, O.	55
Bonders, M.	227	Nikulsins, V.	55
Borglund, E.A.M.	271	Opmanis, M.	125
Borzovs, J.	v	Piho, G.	283
Breslav, A.	197	Polis, E.	257
Būmans, G.	139	Racz, N.	325
Čerāns, K.	80, 125, 139	Rencis, E.	80, 94
Cerina-Berzina, J.	169	Rikacovs, S.	80
Čyras, V.	408	Roost, M.	283
Diebelis, E.	309	Savičienė, L.	408
Dumas, M.	339	Seufert, A.	325
Ebert, J.	19	Skadiņš, R.	153
Grabis, J.	227	Śmiałek, M.	37
Haav, H.-M.	3	Sostaks, A.	66
Kaljuvee, A.	3	Stipravietis, P.	183
Kampars, J.	227	Tepandi, J.	283
Karnitis, G.	169	Vaira, G.	422
Kirikova, M.	v	Vajakas, T.	3
Kornijenko, J.	55	Veski, A.	394
Kozlovics, S.	80	Vohandu, L.	394
Kravcevs, M.	240	Walter, T.	19
Kurasova, O.	422	Weippl, E.	325
Lace, L.	169	Ziema, M.	183
Lapin, K.	408	Zuters, J.	369
Lepmets, M.	296		

This page intentionally left blank